Tecniche avanzate in Assembler con lo ZX SPECTRUM

S. Nicholls

Tecniche avanzate in Assembler con lo ZX SPECTRUM

Tecniche avanzate in Assembler con lo ZX SPECTRUM

S. Nicholls

McGRAW-HILL Book Company GmbH

Amburgo · New York · St Louis · San Francisco · Auckland · Bogotá · Città del Guatemala · Città del Messico · Johannesburg · Lisbona · Londra · Madrid · Montreal · Nuova Delhi · Panama · Parigi · San Juan · San Paolo · Singapore · Sydney · Tokyo · Toronto

Titolo originale: *Assembly Language for Arcade Games and other Fast Spectrum Programs*

Realizzazione editoriale: EDIGEO snc, via Ozanam 10a, 20129 Milano
Traduzione: Giuseppe Zappalà
Grafica di copertina: Valentina Boffa
Composizione e stampa: Litovelox, Trento

ISBN 88-7700-010-4

1ª edizione Novembre 1984

A mia moglie Fran

Indice

Prefazione

Pensavate di aver letto tutto il leggibile, avendo spulciato tutti i manuali sull'Assembler dello Z80, ma proprio nel momento in cui avevate raggiunto una discreta conoscenza delle istruzioni avete capito che volevate saperne di più! I manuali infatti danno una sommaria descrizione di che cos'è e come funziona il codice macchina limitandosi ad alcuni esempi di somme e sottrazioni, ma lasciano il lettore alle prese con i casi concreti da affrontare e risolvere.
Questo è il libro che cercavate, che vi prende per mano dove gli altri vi hanno lasciato e vi mostra come sfruttare la conoscenza delle routine dello Spectrum per produrre programmi perfettamente funzionanti scritti interamente in codice macchina.
Esamineremo varie routine, da quelle più facili, di stampa, a quelle più complesse, che operano su numeri in virgola mobile. L'ultimo capitolo vi mostrerà infine come mettere a frutto le conoscenze acquisite, convertendo in codice macchina un programma BASIC.
I listati del codice macchina sono stati ottenuti usando lo *ZX Spectrum Machine Code Assembler*. Non mi stancherò di ripetere che questo programma è uno strumento indispensabile per chiunque abbia intenzione di realizzare programmi di un certo livello. Esso evita la noiosissima ricerca sulle tabelle dei codici operativi e vi permette di modificare velocemente il programma, fino al suo perfetto funzionamento.
È disponibile il nastro *Arcade Games Routines* contenente tutti i programmi (tanto in BASIC che in codice macchina) presentati su questo libro.

Nota all'edizione italiana

Questo libro è dedicato ai più esperti utenti dello Spectrum desiderosi di approfondire ulteriormente le proprie conoscenze della macchina e delle tecniche avanzate di programmazione. Il libro comunque, grazie alla sua chiarezza espositiva è anche alla portata dei neofiti. I programmi qui presentati sono perfettamente ed immediatamente funzionanti.
Essi sono stati concepiti per l'uso con lo Spectrum in versione base, collegato al registratore ed eventualmente alla stampante ZX. La particolare tecnica di programmazione adottata non permette l'uso dei programmi così come sono nel caso venga sfruttata la ZX Inferface.
In testa ad ogni programma è presente la direttiva

```
org 42000 23760
```

Essa comunica all'Assembler la locazione a partire da cui dovrà immagazzinare provvisoriamente il codice generato (42000) e quella finale di lavoro (23760). Se intendete usare lo Spectrum collegato al solo registratore e, eventualmente, alla stampante ZX, potrete modificare la direttiva in

```
org 23760
```

Il codice sarà posto nella REM iniziale dei vari programmi, che, pur non apparendo per motivi grafici nei listati presentati, andrà predisposto seguendo le istruzioni riportate nel manualetto dello *ZX Spectrum Machine Code Assembler.*
Modificando, invece, la direttiva in

```
org 42000
```

dovrete provvedere a salvare il codice generato come file di tipo "CODE" e a creare un opportuno caricatore.
Lasciando invariata la direttiva, vi sarà possibile assemblare il programma su macchine collegate all'Interface, salvare il codice nel modo sopra descritto, scollegare l'Interface, creare un programma consistente nella sola REM "fittizia", caricandovi quindi sopra il codice preventivamente salvato su cassetta, e salvando (questa volta come file di programma) nuovamente su cassetta.
Potrete eventualmente aggiungere un'opportuna istruzione RANDOMIZE USR 23760 come seconda linea del programma, in modo da ottenere l'esecuzione del codice alla sola pressione del tasto RUN.
Per assemblare i programmi dovrete usare

```
RANDOMIZE USR 58000 per la versione 48K
o
RANDOMIZE USR 26000 per la versione 16K
```

Capitolo 1

Visualizzazione a colori

Anche se la visualizzazione in codice macchina è un lavoro dei più noiosi, che richiede molto tempo, è anche uno dei più importanti. Un buon programma può essere impoverito da una cattiva grafica e da messaggi ambigui; analogamente un programma mediocre può essere migliorato da una buona presentazione.
Nello sviluppo di ogni programma occorre prestare pertanto la massima attenzione alla presentazione grafica.
Lo Spectrum mette a disposizione dell'utente 21 simboli grafici definibili a piacere (UDG o *User-Definable Graphics*), che quindi potrete sfruttare pienamente. Se per esempio state sviluppando un gioco di invasori spaziali, non limitatevi alla definizione delle astronavi, delle bombe e dei colpi, ma usate gli altri caratteri disponibili per personalizzare la visualizzazione del punteggio o per creare suggestivi sfondi. Usate i colori con accortezza: non accostate rosso e magenta (o bianco e giallo) e controllate, con un televisore in bianco e nero, l'effetto ottenuto nella visualizzazione monocromatica dei vostri grafici, poiché l'utilizzatore del programma può non disporre di un televisore a colori.
Poiché ogni programma richiede, in un modo o in un altro, la visualizzazione di informazioni, in questo capitolo esamineremo i metodi di conversione dell'istruzione BASIC PRINT in codice macchina.
Solitamente, la prima istruzione predispone i valori di BORDER, PAPER e INK, e genera un CLS per visualizzare questi colori sullo schermo.
Il colore di BORDER (cornice) è probabilmente il più facile da definire in codice macchina e richiede solo cinque byte. Si provvede a caricare il registro A col numero del colore richiesto, e si chiama quindi la routine residente nella ROM all'indirizzo 8859 (in decimale, equivalente a 229B esa-

decimale). Questa routine inizializza lo schermo col colore di BORDER richiesto e immagazzina il valore attribuito al colore nella variabile di sistema BORDCR 23624 (in effetti il valore immagazzinato è uguale ad otto volte il codice del colore) nei bit 5-3. Gli altri bit sono usati per immagazzinare i parametri di INK, BRIGHT e FLASH per la parte inferiore dello schermo (le linee di input).
La versione in codice macchina per l'inizializzazione di BORDER al valore 4 (verde) è illustrata nel programma 1.1.

Programma 1.1

```
org 42000 23760

10 BORDER 4
20 REM verde

23760 3E 04        ld a,4

CALL BORDER SET

23762 CD 9B 22     call 8859
23765 C9           ret
```

I colori di PAPER (sfondo) e INK (caratteri) sono conservati nella variabile di sistema ATTR-P (23693). I bit 0-2 contengono il colore di INK e i bit 3-5 quello di PAPER. I bit 6 e 7 sono sfruttati per indicare FLASH e BRIGHT. Con FLASH 0 si resetta il bit 7, che viene settato da FLASH 1; BRIGHT 0 resetta il bit 6 che viene settato da BRIGHT 1.
Nel programma 1.2 possiamo vedere come eseguire PAPER 3:INK 1:CLS; da notare che ATTR-P (23693) può essere considerata pari a IY+83.
Il registro IY viene usato come puntatore alle variabili di sistema e contiene il valore 23610. Ad esempio, l'indirizzo della cella 23609 può essere considerato come (IY−1) o (IY+255).

Programma 1.2

```
org 42000 23760
 5 REM predispone i colori
    permanenti di schermo
10 PAPER 3  INK 1  CLS

23760 3E 19        ld a,25

Ld (ATTR-P),colori

23762 FD 36 53 19 ld (iy+83),25
```

```
Call apri canale 2

23766 3E 02       ld a,2
23768 CD 01 16    call 5633

call CLS

23771 CD 6B 0D    call 3435
23774 C9          ret
```

Il comando CLS esegue un CALL 3435 che pulisce lo schermo e assegna i parametri in modo analogo al comando CLS del BASIC; noterete che prima di eseguire CALL 3435, si è provveduto a eseguire CALL 5633. Ciò è molto importante e sarà usato diverse volte in questo capitolo. La chiamata alla subroutine 5633 è necessaria per aprire il canale 2, permettendo così alle informazioni di venire visualizzate nella parte superiore dello schermo, cioè nelle linee da 0 a 21. Presentati i parametri permanenti di schermo, possiamo ora passare alla vera e propria visualizzazione dei nostri dati.
Il programma 1.3 illustra il metodo di visualizzazione di un carattere nella successiva posizione di schermo (0,0 se è stato eseguito precedentemente un CLS). Come potete vedere è necessario aprire il canale 2 e immagazzinare nel registro A il codice del carattere da visualizzare.
La routine di visualizzazione di un carattere viene chiamata per mezzo di RST 16, che a sua volta chiama la routine 5618 che visualizza il carattere contenuto nel registro A, e muove di un posto la posizione di visualizzazione.

Programma 1.3

```
org 42000 23760
 5 REM visualizza un carattere
10 PRINT "A"

Apri canale 2

23760 3E 02       ld a,2
23762 CD 01 16    call 5633

PONE NELL'ACCUMULATORE IL CODI-
CE DEL CARATTERE "A"

23765 3E 41       ld a,65

CALL PRINT

23767 D7          rst 16
23768 C9          ret
```

Il programma 1.4 è un'estensione del precedente per permettere la visualizzazione di una breve parola. Viene aperto il canale 2 e l'accumulatore viene caricato di un carattere per volta fino alla visualizzazione dell'intera parola.

Programma 1.4

```
org 42000 23760
 5 REM visualizza una stringa
    di caratteri
10 PRINT "HELLO"

Apri canale 2

23760 3E 02       ld a,2
23762 CD 01 16    call 5633
23765 3E 48       ld a,72
H
23767 D7          rst 16
23768 3E 45       ld a,69
E
23770 D7          rst 16
23771 3E 4C       ld a,76
L
23773 D7          rst 16
23774 3E 4C       ld a,76
L

23776 D7          rst 16
23777 3E 4F       ld a,79
O
23779 D7          rst 16
23780 C9          ret
```

Il metodo appena esposto richiede però tempo e memoria per visualizzare una parola lunga o una frase; sarà pertanto opportuno sfruttare un sistema più efficiente.
Come si può constatare esaminando il programma 1.5, abbiamo a disposizione varie possibilità. L'idea principale consiste nell'immagazzinare il messaggio come DATA in modo che sia letto un carattere alla volta fino alla sua completa visualizzazione.

Programma 1.5

```
org 42000 23760
 5 REM visualizza una
 LUNGA stringa
10 PRINT "HELP ME"
```

```
23760 3E 02        ld a,2
23762 CD 01 16     call 5633
23765 11 E4 5C     ld de,DATA
23768 01 07 00     ld bc,7

Lunghezza stringa in BC

L1
23771 78           ld a,b
23772 B1           or c
23773 0B           dec bc
23774 C8           ret z
23775 1A           ld a,(de)
23776 13           inc de
23777 D7           rst 16

23778 18 F7        jr L1
DATA
defb 72 69 76 80 32 77 69
```

Nella versione 1 il registro DE contiene l'indirizzo iniziale dei DATA e viene sfruttato come puntatore ai singoli caratteri; la lunghezza della stringa è immagazzinata nel registro BC; la visualizzazione si ottiene chiamando RST 16 per ogni carattere, finché BC=0.

```
org 42000 23760
23760 3E 02        ld a,2
23762 CD 01 16     call 5633
23765 11 DF 5C     ld de,DATA
23768 01 07 00     ld bc,7
CALL PRINT STRING

23771 CD 3C 20     call 8252
23774 C9           ret
DATA
defb 72 69 76 80 32 77 69
```

La versione 2 è analoga alla versione 1: sfrutta infatti la routine 8252, che è una copia della routine della versione 1 dall'indirizzo 23771 all'indirizzo 23778.

```
org 42000 23760
23760 3E 02        ld a,2
23762 CD 01 16     call 5633
23765 11 E5 5C     ld de,DATA
Loop
23768 1A           ld a,(de)
```

```
23769 CB 7F        bit 7,a
23771 20 04        jr nz,END
23773 D7           rst 16
23774 13           inc de
23775 18 F7        jr Loop
END
23777 CB BF        res 7,a
23779 D7           rst 16
23780 C9           ret
DATA
defb 72 69 76 80 32 77 197
```

La versione 3 può essere sfruttata per la stampa di stringhe di qualunque lunghezza, ma aventi caratteri con codici compresi fra 0 e 127. L'ultimo carattere è usato come indicatore per segnalare la fine della stringa; esso ha il numero 128 sommato al proprio codice, il che equivale al bit 7 posto a 1; prima della stampa del carattere sarà pertanto necessario controllare se il bit 7 è a 0. Se questa condizione è verificata il ciclo di stampa si ripete; in caso contrario viene eseguito un salto alla routine END, che resetta il bit 7 dell'accumulatore, stampa il carattere e continua col programma, incontrando, in questo caso, l'istruzione RET.
La versione 3 può essere sfruttata come subroutine PRINT in un programma in cui svariati messaggi possono essere immagazzinati insieme, separati solo dagli indicatori di fine; è quindi necessario caricare solamente l'indirizzo iniziale del messaggio nella coppia di registri DE e chiamare la subroutine PRINT. Una routine simile ma più complessa è usata dallo Spectrum per la visualizzazione delle parole chiave e dei messaggi di errore; l'indirizzo di questa routine è 3082.
I messaggi di errore da 1 a R elencati nell'appendice B del manuale Spectrum possono essere facilmente visualizzati usando l'istruzione RST 8. Il byte che segue RST 8 punta al messaggio d'errore richiesto. Se, per esempio, desiderate ottenere il messaggio K INVALID COLOUR, il messaggio d'errore numero 21, usate:

```
RST 8
defb 19 (numero del messaggio −2)
```

Non è necessario includere un'istruzione RET, perché la ROM dopo aver visualizzato l'errore cederà automaticamente il controllo al BASIC.
Possiamo ora passare alla visualizzazione di caratteri o stringhe in determinate linee e colonne, come nel comando BASIC PRINT AT. Come potete notare consultando l'appendice A del manuale Spectrum i codici da 6 a 23 sono codici di controllo; in particolare il codice 22 rappresenta AT. L'istruzione RST 16 riconosce il codice 22 come PRINT AT e sfrutta i due successivi byte di DATA come coordinate x,y che identificano la posizio-

ne della prima lettera del messaggio da visualizzare. Il programma 1.6 usa questa caratteristica usando defb 22,3,5 come PRINT AT 3,5.

Programma 1.6

```
org 42000 23760

10 REM visualizza una stringa
 in 3,5
20 PRINT AT 3,5 "Grazie mille"

23760 3E 02        ld a,2
23762 CD 01 16     call 5633
23765 11 DF 5C     ld de,DATA
23768 01 0F 00     ld bc,15
23771 CD 3C 20     call 8252
23774 C9           ret
DATA

AT 3 5

defb 22 3 5
defs Grazie mille
```

Lo stesso principio può essere usato per TAB (codice 23). Notate anche che ENTER (codice 13) produrrà lo stesso effetto di NEWLINE.
Dovremmo ora essere in grado di visualizzare qualunque messaggio in qualunque posizione nelle linee da 0 a 21 usando i colori permanenti. Ci resta ora, per il completo controllo dello schermo, di visualizzare messaggi in colori locali, sfruttando i modi BRIGHT, FLASH, OVER, INVERSE.
Il programma 1.7 mostra come ottenere quanto desiderato e utilizza lo stesso principio del programma 1.6, immagazzinando in DATA i codici di PAPER, INK, BRIGHT..., con il byte che segue il codice di controllo dedicato alla specifica dei parametri.

Programma 1.7

```
org 42000 23760

10 REM visualizzazione a colori
20 PRINT AT 4,7 PAPER 2
INK 4 FLASH 1 BRIGHT 1 OVER 1
INVERSE 1 "Riesci a leggere?"

23760 3E 02        ld a,2
23762 CD 01 16     call 5633
23765 11 E2 5C     ld de,DATA
```

```
23768 01 24 00     ld bc,36
23771 CD 3C 20     call 8252

RESETTA I COLORI PERMANENTI
E PONE BRIGHT 0 FLASH 0

23774 CD 4D 0D     call 3405
23777 C9           ret
DATA
defb 22 4 7 17 2 16 4 18 1

defb 19 1 21 1 20 1
defs Riesci a leggere?
defb 21 0 20 0
```

Una volta usati i colori locali, è necessario ritornare ai colori permanenti per le successive visualizzazioni. Dopo l'uso di OVER e INVERSE, ciò è possibile eseguendo RST 16; per quanto riguarda PAPER, INK, BRIGTH e FLASH, si potrà chiamare la routine 3405, che copia il valore di ATTR P (23693) in ATTR T (23695).
Consideriamo ora la visualizzazione nella parte inferiore dello schermo, cioè nelle linee 22 e 23. Il metodo è uguale a quello appena descritto, ma richiede l'apertura di un diverso canale prima dell'esecuzione di RST 16. Questo procedimento è illustrato nel programma 1.8.

Programma 1.8

```
org 42000 23760

10 REM visualizzazione sulle
 linee di INPUT
20 PRINT #0 AT 0,10
 "linea di INPUT 0"
 AT 1,10 "linea di INPUT 1"


Apre il canale logico -3

23760 3E FD        ld a,253
23762 CD 01 16     call 5633
23765 11 EA 5C     ld de,DATA
23768 01 26 00     ld bc,38
23771 CD 3C 20     call 8252

WAIT

L1
```

```
23774 76          halt

If tasto premuto then
BIT 5,FLAGS posto a 1 (SET)

23775 FD CB 01 6E bit 5,(iy+1)
23779 28 F9       jr z,L1
23781 FD CB 01 AE res 5,(iy+1)
23785 C9          ret
DATA
defb 22 0 10
defs Linea di INPUT 0
defb 22 1 10
defs Linea di INPUT 1
```

Abbiamo sfruttato il canale −3 (253).
Il programma BASIC è interessante in quanto illustra una caratteristica non menzionata nel manuale: PRINT#0 visualizza sulle linee di input. L'istruzione PAUSE 0 è necessaria per fermare il programma evitando così la visualizzazione del messaggio OK, che cancellerebbe quanto appena visualizzato.
Anche la routine in codice macchina richiede un equivalente di PAUSE 0. Questo è ottenuto testando il bit 5 di FLAGS (23611) o (IY+1) dopo l'esecuzione dell'istruzione HALT. Questa istruzione attende in effetti la successiva routine KEYSCAN e, se viene premuto un tasto, setta il bit 5 della variabile FLAGS. Restiamo quindi in un loop di attesa fino a quando il bit 5 viene settato, ritornando poi al BASIC. Da notare che la linea 22 viene considerata linea 0 e la linea 23 linea 1.
Una routine molto utile per eliminare quanto visualizzato sulle linee di ingresso, lasciando intatto il resto dello schermo, inizia all'indirizzo 3652. Si tratta di una routine che cancella un numero specificato di linee, contate dal fondo dello schermo. Il numero contenuto nel registro B deve essere maggiore di 0 e minore di 25; si provvede quindi a chiamare la routine 3652. I colori usati per cancellare le linee sono quelli contenuti in ATTR P. È possibile cancellare le linee di ingresso con le istruzioni:

```
LD B, 2
CALL 3652
(RETURN)
```

Prima di passare al prossimo capitolo è importante richiamare l'attenzione sui caratteri grafici definibili dall'utente. Se avete scritto un programma con l'intenzione di eseguirlo sia su Spectrum da 16K che da 48K, gli UDG devono essere correttamente immagazzinati in ogni versione. Il modo più semplice per farlo consiste nel creare gli UDG iniziando all'indi-

rizzo 32600, cioè per lo Spectrum 16K, sfruttando quindi il programma 1.9 per rilocare gli UDG, se necessario, all'indirizzo contenuto nella variabile di sistema UDG (23675).

Programma 1.9

```
  10 REM Riloca gli UDG creati per lo
Spectrum 16K per l'uso   con la versione 48K
  20  LET de = PEEK 23675 + 256 *PEEK 23676
  30 LET hl = 32600
  40 LET bc = 166
  50 POKE de,PEEK hl
  55 REM !RILOCA GLI UDG;!
  60 LET hl = hl + 1
  70 LET de = de + 1
  80 LET bc = bc + 1
  90 IF bc <> 0  THEN  GO TO 50
 100 REM return


org 42000 23760

VERSIONE IN CODICE MACCHINA

RILOCA GLI UDG

23760 ED 5B 7B 5C ld de,(23675)
23764 21 58 7F    ld hl,32600
23767 01 A8 00    ld bc,168
23770 ED B0       ldir
23772 C9          ret
```

Capitolo

PLOT, DRAW e CIRCLE

PLOT

Il comando BASIC PLOT x,y controlla l'accensione del pixel situato in colonna x, riga y col colore di INK. Il pixel in posizione 0,0 è quello nell'angolo in basso a sinistra dello schermo; l'angolo in alto a destra è identificato dalla coppia 175,255.
In codice macchina disponiamo di due utili *entry point* alla routine PLOT: il primo di questi, che è probabilmente il più utile e facile da usare, viene sfruttato tramite CALL 8933. Prima di chiamare la routine occorre immagazzinare nel registro B il valore di y (variabile fra 0 e 175) e nel registro C il valore di x (0–255). Il programma 2.1 esegue il semplice comando BASIC PLOT 75,125.

Programma 2.1

```
org 42000 23760

10 REM plot x,y
20 PLOT 75,175

23760 06 7D       ld b,125
23762 0E 4B       ld c,75
23764 CD E5 22    call 8933
23767 C9          ret
```

È più facile comprendere il programma usando la notazione decimale in-

vece di quella esadecimale e caricando i registri B e C separatamente. Sarebbe ovviamente stato possibile risparmiare memoria usando LD BC, 32075 ma questo avrebbe potuto generare confusione.
Il secondo entry point è in 8924; esso richiede che vengano caricati i valori di x e y nello *stack* (catasta) di calcolo, immagazzinando in cima il valore di y. Il procedimento viene illustrato dal programma 2.2, che esegue lo stesso PLOT 75,125.

Programma 2.2

```
org 42000 23760
23760 3E 4B        ld a,75

STACK 75

23762 CD 28 2D     call 11560
23765 3E 7D        ld a,125

STACK 175

23767 CD 28 2D     call 11560

CALL PLOT

23770 CD DC 22     call 8924
23773 C9           ret
```

La routine presente nella ROM all'indirizzo 11560 viene usata per immagazzinare in cima allo stack di calcolo il valore contenuto nel registro A. Sfruttando la routine PLOT all'indirizzo 8924, si provvede a rimuovere gli ultimi due valori immagazzinati nello stack e ad "accendere" il punto così identificato. Lo stack viene ovviamente ridotto di due valori. L'uso della catasta sarà discusso più approfonditamente nel programma 2.5.
Il secondo entry point è utile se lo stack è già stato sfruttato per la manipolazione di una formula per tracciare grafici.
Il programma 2.3 mostra come ottenere PLOT OVER 1 x,y.

Programma 2.3

```
org 42000 23760

10 REM plot OVER 1 x,y
20 PLOT OVER 1 25,40

PREDISPONE OVER 1
PONE A 1 I BIT 0 & 1 DI P-FLAG
```

```
23760 3E 03       ld a,3
23762 FD 77 57    ld (iy+87),a
23765 06 28       ld b,40
23767 0E 19       ld c,25
23769 CD E5 22    call 8933

PREDISPONE OVER 0
AZZERA I BIT 0 & 1 DI P-FLAG

23772 AF          xor a
23773 FD 77 57    ld (iy+87),a
23776 C9          ret
```

Il programma pone a uno i bit 0 e 1 di P FLAG (IY+87) prima di eseguire il CALL PLOT e quindi li resetta. Trascurando questo accorgimento, ogni successivo tracciato sarà eseguito come OVER 1 invece che come OVER 0. L'istruzione PLOT INVERSE è formalmente identica a PLOT OVER; richiede però l'azione sui bit 2 e 3 della variabile P FLAG, che assume quindi il valore 12.
Sfruttando l'entry point 8933 possiamo scrivere un programma in grado di tracciare un carattere in qualunque punto dello schermo. Le uniche limitazioni sono date dal codice del carattere che deve essere compreso tra 32 (spazio) e 127 (copyright ©): è cioè possibile visualizzare solo caratteri la cui forma, rappresentata da otto byte, sia contenuta nella ROM del generatore di caratteri.

Programma 2.4

```
org 42000 23760

TRACCIA UN CARATTERE
CODICE CARATTERE >=32 AND <=127

POSIZIONE DI TRACCIAMENTO
IMMAGAZZINATA COME DATA XY

Pone in L il codice  CARATTERE

23760 2E 7F       ld l,127
23762 26 00       ld h,0

Moltiplica x8

23764 29          add hl,hl
23765 29          add hl,hl
23766 29          add hl,hl
```

```
Somma a (23606) per trovare
l'inizio del carattere

23767 ED 5B 36 5C ld de,(23606)
23771 19          add hl,de

Contatore di Byte

23772 06 08       ld b,8
L3
Salva il puntatore HL

23774 E5          push hl

Contatore di bit

23775 0E 08       ld c,8
23777 7E          ld a,(hl)
L2
23778 C5          push bc

Acquisisce la posizione x,y


23779 ED 4B 0E 5D ld bc,(XY)

 BIT SET ?

23783 17          rla
23784 F5          push af
23785 30 05       jr nc,L1

If SET then PLOT

23787 C5          push bc
23788 CD E5 22    call 8933
23791 C1          pop bc
L1

Muove PLOT a x+1,y

23792 0C          inc c
23793 ED 43 0E 5D ld (XY),bc
23797 F1          pop af

23798 C1          pop bc
23799 0D          dec c
23800 20 E8       jr nz,L2
23802 C5          push bc

Ripristina PLOT a x-8,y-1
```

```
23803 ED 4B 0E 5D ld bc,(XY)
23807 05          dec b
23808 3E F8       ld a,248
23810 81          add a,c
23811 4F          ld c,a
23812 ED 43 0E 5D ld (XY),bc
23816 C1          pop bc
23817 E1          pop hl
23818 23          inc hl

Ripete finche' cont.BYTE = 0

23819 10 D1       djnz L3
23821 C9          ret

XY
defb 100 20
```

Come potete vedere nel programma 2.4, il codice del carattere da visualizzare viene immagazzinato nel registro L, moltiplicato per otto e sommato al valore contenuto nella variabile di sistema (23606) per trovare l'indirizzo iniziale degli otto byte che contengono la forma del carattere. La coppia di registri HL viene quindi usata come puntatore al byte del carattere sotto esame; la coppia di registri BC viene sfruttata come contatore per la matrice da otto per otto bit che compone il carattere, mentre il registro A contiene il byte sotto esame.
Una istruzione RLA viene eseguita otto volte per ogni byte e se un bit è a uno, cioè, se dopo l'esecuzione dell'istruzione RLA il flag di CARRY è a uno, viene acceso il corrispondente pixel.
Le coordinate x, y vengono aggiornate dopo l'esecuzione di ogni istruzione RLA. Le coordinate iniziali x,y identificano il pixel superiore sinistro del carattere sotto processo. Per restare sullo schermo sarà necessario che il valore di y vari fra 7 e 175. Il valore di x potrà essere nel campo da 0 a 255 e ogni tentativo di assegnare a x un valore superiore a 255 lo resetterà a 0, pertanto: se C = 255, l'esecuzione dell'istruzione "inc C" restituirà C = 0.
È possibile modificare agevolmente il programma per visualizzare caratteri ruotati di 90 gradi in senso antiorario.

Programma 2.5

```
org 42000 23760

RUOTA IL CARATTERE 90 GRADI
IN SENSO ANTIORARIO
```

```
Pone in L il codice CARATTERE

23760 2E 60        ld l,96
23762 26 00        ld h,0

Moltiplica x 8

23764 29           add hl,hl
23765 29           add hl,hl
23766 29           add hl,hl

Somma a (23606) per trovare
l'inizio del carattere

23767 ED 5B 36 5C ld de,(23606)

23771 19           add hl,de

Contatore di byte

23772 06 08        ld b,8
L3
Salva il puntatore HL

23774 E5           push hl

Contatore di bit

23775 0E 08        ld c,8
23777 7E           ld a,(hl)
L2
23778 C5           push bc

Acquisisce la posizione x,y

23779 ED 4B 0E 5D ld bc,(XY)


 BIT SET ?

23783 17           rla
23784 F5           push af
23785 30 05        jr nc,L1

If SET then PLOT

23787 C5           push bc
23788 CD E5 22     call 8933
23791 C1           pop bc
L1
```

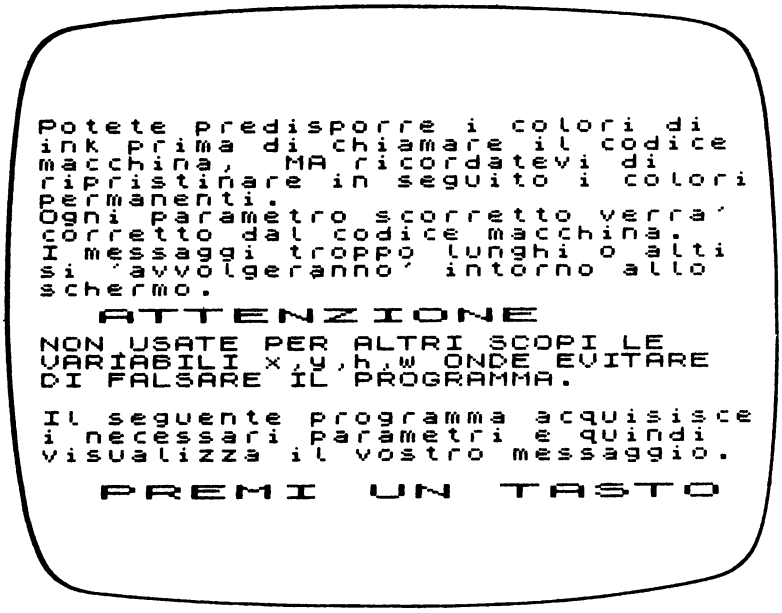
```
Muove PLOT a x,y+1

23792 04          inc b
23793 ED 43 0E 5D ld (XY),bc
23797 F1          pop af
23798 C1          pop bc
23799 0D          dec c
23800 20 E8       jr nz,L2

23802 C5          push bc

Ripristina PLOT a x+1,y-8

23803 ED 4B 0E 5D ld bc,(XY)
23807 0C          inc c
23808 3E F8       ld a,248
23810 80          add a,b
23811 47          ld b,a
23812 ED 43 0E 5D ld (XY),bc
23816 C1          pop bc
23817 E1          pop hl
23818 23          inc hl

Ripete finche' cont. BYTE = 0

23819 10 D1       djnz L3
23821 C9          ret
XY
defb 100 20
```

Nel programma 2.5, le coordinate x,y vengono aggiornate per fornire un tracciamento verticale, anziché orizzontale come nel programma 2.4; le coordinate x,y identificano pertanto il pixel inferiore sinistro e il campo di variazione ammesso per y è quindi da 0 a 167.
Abbiamo esaminato il metodo di tracciamento di un carattere; possiamo ora passare al nostro primo programma impegnativo per visualizzare un messaggio in qualunque posizione dello schermo e, per aumentare l'utilità del programma, faremo in modo di poter scegliere l'altezza e la larghezza del carattere. (Una copia del programma è disponibile su nastro sotto il nome di PLOTDEMO).
Poiché questo programma è particolarmente utile da sfruttare come routine in un programma BASIC, vi ho inserito una routine di acquisizione dati, così da poter inizializzare i parametri altezza/larghezza, x/y e il messaggio sfruttando variabili BASIC, evitando quindi la necessità di introdurli nel codice macchina. Se si desidera sfruttare la routine insieme ad un programma in codice macchina, la routine di acquisizione diventa inutile e può tranquillamente essere omessa e i parametri possono essere co-

piati direttamente nel buffer di stampa. Il programma BASIC è autoesplicante e permette di eseguire esperimenti con diversi valori dei parametri. Il codice macchina è un'estensione del programma 2.4. I valori di x,y,h e w vengono ottenuti per mezzo della routine di acquisizione. Questa routine scandisce le variabili fino al ritrovamento del CODE (codice) della variabile richiesta e ne immagazzina il valore nel buffer di stampa. Eseguita questa operazione, si ricerca e si immagazzina il valore di LEN a$ e il messaggio viene infine copiato nel buffer di stampa. La routine PLOT include una istruzione di CATCH-ALL in grado di portare a valori accettabili qualunque parametro errato. La routine consiste di cinque loop annidati:

(L1) traccia un bit SET per una larghezza w
(L2) ripete L1 per otto bit
(L3) ripete per h (altezza) volte
(L4) ripete per otto byte
(L5) ripete fino alla fine del messaggio

Prestate particolare attenzione ai commenti sulle variabili dei loop. Per rendere le videate più leggibili sono stati inseriti appositi spazi.



Programma 2.6

```
    5 OVER 0: INK 0: PAPER 6: BORDER 3: CLS
  10 LET x=10: LET y=79: LET h=5: LET w=2: LE
T a$="FERMA IL NASTRO"
  15 INK 2: RANDOMIZE USR 32393
  18 PAUSE 100
  20 LET w=8: LET y=150: LET x=0: LET a$="PLO
T"
  30 INK 1: RANDOMIZE USR 32393
  35 LET a$="____": LET y=140
  36 INK 1: RANDOMIZE USR 32393
  40 FLASH 1: GO SUB 8000
  50 FLASH 0
  60 PAUSE 0
  70 PAPER 7: INK 0: BORDER 7: CLS
  80 LET x=32: LET y=175: LET h=1: LET w=2: L
ET a$="ISTRUZIONI"
  90 INK 2: RANDOMIZE USR 32393: INK 0
 100 PRINT AT 2,0;"Questo  programma  dimostr
ativo vi permettera' di 'scrivere' in qualunq
ue parte dello schermo i vostri   'messaggi'
  usando le variabili BASIC per predisporne i
 parametri."
 110 PRINT '"I  parametri  dei  caratteri  si
predispongono cosi' :-"
```

```
 120 PRINT ''"Altezza   : LET h=1 to h=22"'"L
arghezza : LET w=1 to w=32"
 130 PRINT "asse x     : LET x=0 to 255"'"asse
 y     : LET y=0 to 175"
 140 PRINT "Messaggio : LET a$=";CHR$ 34;"***
******";CHR$ 34
 150 PRINT "Plot       : RANDOMIZE USR 32393"
 160 GO SUB 8000
 170 IF INKEY$<>"" THEN  GO TO 170
 175 IF INKEY$="" THEN  GO TO 175
 178 CLS
 180 PRINT "Potete predisporre i colori di  i
nk prima di chiamare il codice macchina,  MA
 ricordatevi di    ripristinare in seguito i c
oloripermanenti."
 190 PRINT "Ogni parametro scorretto verra' c
orretto dal codice macchina.   I messaggi tro
ppo lunghi o alti si 'avvolgeranno' intorno a
llo  schermo."
 200 LET a$="ATTENZIONE": LET y=91: INK 2: RA
NDOMIZE USR 32393: INK 0
 210 PRINT AT 12,0;"NON USATE PER ALTRI SCOPI
 LE    VARIABILI x,y,h,w ONDE EVITARE  DI FAL
SARE IL PROGRAMMA."
 220 PRINT '"Il seguente programma acquisisce
i necessari parametri e quindi  visualizza il
 vostro messaggio."
 230 GO SUB 8000
 235 IF INKEY$<>"" THEN  GO TO 235
 240 IF INKEY$="" THEN  GO TO 240
 242 CLS : BORDER 4
 245 PLOT 0,121: DRAW 255,0: DRAW 0,-73: DRAW
 -255,0: DRAW 0,73
 250 PRINT AT 8,0;"ALTEZZA    1-22 ?     __","
LARGHEZZA  1-32 ?     __","asse X     0-255 ?
   ___","asse Y     0-175 ?   ___","MESSAGGIO
?","_____INK
 0-9 ?",,
 255 LET z$="
"
 257 LET v$="
"
 260 PRINT  OVER 1;AT 8,0;z$
 262 INPUT h: PRINT AT 8,21;h;" "; OVER 1;AT
8,0;v$
 265 PRINT  OVER 1;AT 9,0;z$: INPUT w: PRINT
AT 9,21;w;" "; OVER 1;AT 9,0;v$
 270 PRINT  OVER 1;AT 10,0;z$: INPUT x: PRINT
 AT 10,21;x;"  "; OVER 1;AT 10,0;v$
 272 PRINT  OVER 1;AT 11,0;z$: INPUT y: PRINT
```

```
 AT 11,21;y;"  "; OVER 1;AT 11,0;v$
 273 PRINT  OVER 1;AT 12,0;z$;AT 13,0;z$: INP
UT a$: PRINT AT 13,0;a$;: FOR k=LEN a$ TO 31:
 PRINT " ";: NEXT k: PRINT  OVER 1;AT 12,0;v$
; OVER 1;AT 13,0;v$
 275 PRINT  OVER 1;AT 14,0;z$: INPUT i: PRINT
 AT 14,0; INK i; OVER 1;v$
 280 PAUSE 50
 290 CLS : INK I: RANDOMIZE USR 32393: INK 0
 300 PRINT #0;AT 0,0;"PREMI Q PER FERMARE";AT
 1,0;"QUALUNQUE ALTRO TASTO            PER UN'
ALTRA ESECUZIONE"
 310 PAUSE 0: IF INKEY$="q" OR INKEY$="Q" THE
N  STOP
 320 CLS : GO TO 245
8000 LET h=1: LET w=2: LET x=20: LET y=15: LE
T a$="PREMI UN TASTO": INK 2: RANDOMIZE USR 3
2393: INK 0
8010 RETURN
9000 STOP
9800 CLEAR 32334: LOAD ""CODE : GO TO 5


org 42000 32335
FIND
32335 2A 4B 5C    ld hl,(23627)
L2
32338 3A B0 5C    ld a,(23728)
32341 BE          cp (hl)
32342 C8          ret z
32343 CB 6E       bit 5,(hl)
32345 20 08       jr nz,L1
32347 23          inc hl
32348 5E          ld e,(hl)
32349 23          inc hl
32350 56          ld d,(hl)
32351 19          add hl,de
32352 23          inc hl
32353 18 EF       jr L2
L1
32355 CB 76       bit 6,(hl)
32357 20 0C       jr nz,L3
32359 23          inc hl

32360 7E          ld a,(hl)
32361 CB 7F       bit 7,a
32363 28 FA       jr z,-6
32365 11 06 00    ld de,6
32368 19          add hl,de
32369 18 DF       jr L2
```

```
L3
32371 CB 7E        bit 7,(hl)
32373 28 F6        jr z,-10
32375 11 13 00     ld de,19
32378 19           add hl,de
32379 18 D5        jr L2
SET
32381 32 B0 5C     ld (23728),a
32384 CD 4F 7E     call FIND
32387 23           inc hl
32388 23           inc hl
32389 23           inc hl
32390 7E           ld a,(hl)
32391 02           ld (bc),a
32392 C9           ret

START
32393 01 00 5B     ld bc,23296
32396 3E 78        ld a,120
32398 CD 7D 7E     call SET
32401 03           inc bc
32402 3E 79        ld a,121
32404 CD 7D 7E     call SET
32407 03           inc bc
32408 3E 68        ld a,104
32410 CD 7D 7E     call SET
32413 03           inc bc
32414 3E 77        ld a,119
32416 CD 7D 7E     call SET
32419 3E 41        ld a,65
32421 32 B0 5C     ld (23728),a
32424 CD 4F 7E     call FIND
32427 23           inc hl
32428 5E           ld e,(hl)
32429 23           inc hl
32430 56           ld d,(hl)
32431 ED 53 04 5B  ld (23300),de

32435 D5           push de
32436 C1           pop bc
32437 23           inc hl
32438 11 05 5B     ld de,23301
32441 ED B0        ldir
32443 2A 00 5B     ld hl,(23296)
32446 AF           xor a
32447 7C           ld a,h
32448 DE B0        sbc a,176
32450 38 04        jr c,+4
32452 67           ld h,a
32453 22 00 5B     ld (23296),hl
```

```
32456 22 B0 5C     ld (23728),hl
32459 21 05 5B     ld hl,23301
LP5
32462 E5           push hl
32463 7E           ld a,(hl)
32464 26 00        ld h,0
32466 6F           ld l,a
32467 29           add hl,hl
32468 29           add hl,hl

32469 29           add hl,hl
32470 11 00 3C     ld de,15360
32473 19           add hl,de
32474 06 08        ld b,8
LP4
32476 C5           push bc
32477 ED 4B 01 5B  ld bc,(23297)
LP3
32481 7E           ld a,(hl)
32482 E5           push hl
32483 C5           push bc
32484 06 08        ld b,8
LP2
32486 C5           push bc
32487 17           rla
32488 F5           push af
32489 DA F9 7E     jp c,PLOT
32492 2A 03 5B     ld hl,(23299)
32495 3A B0 5C     ld a,(23728)
32498 85           add a,l
32499 32 B0 5C     ld (23728),a

32502 C3 0F 7F     jp SKIP
PLOT
32505 ED 4B B0 5C  ld bc,(23728)
LP1
32509 C5           push bc
32510 ED 4B B0 5C  ld bc,(23728)
32514 C5           push bc
32515 CD E5 22     call 8933
32518 C1           pop bc
32519 0C           inc c
32520 ED 43 B0 5C  ld (23728),bc
32524 C1           pop bc
32525 10 EE        djnz LP1
SKIP
32527 F1           pop af
32528 C1           pop bc
32529 10 D3        djnz LP2
32531 3A 00 5B     ld a,(23296)
```

```
32534 21 B0 5C    ld hl,23728
32537 77          ld (hl),a
32538 23          inc hl

32539 AF          xor a
32540 7E          ld a,(hl)
32541 DE B0       sbc a,176
32543 38 03       jr c,+3
32545 77          ld (hl),a
32546 18 08       jr +8
32548 7E          ld a,(hl)
32549 FE 00       cp 0
32551 20 02       jr nz,+2
32553 36 B0       ld (hl),176
32555 35          dec (hl)
32556 C1          pop bc
32557 E1          pop hl
32558 10 B1       djnz LP3
32560 23          inc hl
32561 C1          pop bc
32562 10 A8       djnz LP4
32564 3A 03 5B    ld a,(23299)
32567 87          add a,a
32568 87          add a,a
32569 87          add a,a

32570 6F          ld l,a
32571 3A B0 5C    ld a,(23728)
32574 85          add a,l
32575 32 00 5B    ld (23296),a
32578 32 B0 5C    ld (23728),a
32581 3A 01 5B    ld a,(23297)
32584 32 B1 5C    ld (23729),a
32587 E1          pop hl
32588 23          inc hl
32589 3A 04 5B    ld a,(23300)
32592 3D          dec a
32593 C8          ret z
32594 32 04 5B    ld (23300),a
32597 C3 CE 7E    jp LP5
```

Il codice macchina del programma 2.6 inizia all'indirizzo 32393. La prima routine, da 32393 a 32418, inizializza la coppia di registri BC come puntatore al buffer di stampa e chiama la subroutine SET per selezionare il codice della variabile di indirizzo 23728. La routine SET chiama a sua volta la routine FIND per trovare il codice delle variabili. Il valore della variabile (0−255) viene ora immagazzinato nell'indirizzo del buffer di stampa puntato dai registri BC. In questo modo le variabili x,y,h e w

vengono consecutivamente acquisite e immagazzinate. La routine da 32419 a 32442 trova il codice della variabile a$, la cui lunghezza (0−255) viene ora immagazzinata; la stringa viene infine copiata. La routine da 32443 a 32458 controlla il valore di y (che, se maggiore di 175, viene portato a −176) e immagazzina di nuovo i parametri x,y in 23296/7 e 23728/9.
Il resto del codice macchina è un'estensione della routine di visualizzazione di caratteri, con loop aggiunti per tracciare il pixel w volte per h linee, provvedendo a un opportuno controllo dopo ogni linea per assicurare che la posizione successiva di PLOT faccia parte dello schermo.
La routine da 32540 a 32554 controlla la successiva posizione di PLOT e genera un effetto di avvolgimento continuo se la posizione è al di fuori dell'estremità inferiore dello schermo. Senza andare più a fondo nelle routine CALCULATOR, possiamo esaminare il programma che sfrutta PLOT all'indirizzo 8924.
Il programma 2.7 mostra come visualizzare una sinusoide sfruttando CALCULATOR per manipolare la formula:

88+80*SIN (A/128*PI)

Programma 2.7

```
org 42000 23760
10 FOR A=0 TO 255
20 PLOT A,88+80*SIN (A/128*PI)
30 NEXT A

LET A=0

23760 AF          xor a
LOOP
SALVA A
23761 F5          push af
STACK A
23762 CD 28 2D    call 11560
23765 3E 58       ld a,88
23767 CD 28 2D    call 11560
23770 3E 50       ld a,80
23772 CD 28 2D    call 11560
23775 F1          pop af
23776 F5          push af
23777 CD 28 2D    call 11560

23780 3E 80       ld a,128
23782 CD 28 2D    call 11560

STACK 88 80 A 128
USA CALCULATOR
```

```
23785 EF          rst 40
Divide i 2 valori in cima
allo stack (A/128)
defb 5
Stack PI/2
defb 163
Duplica i 2 valori in cima
allo stack
defb 49
Somma i 2 valori in cima
allo stack (PI)
defb 15
Moltiplica i 2 valori in cima
allo stack
(A/128*PI)

defb 4
SENO del valore in cima
allo stack: SIN (A/128*PI)
defb 31
Moltiplica i 2 valori in cima
allo stack: 80*SIN (A/128*PI)
defb 4
Somma i 2 valori in cima
allo stack: 88+80*SIN (A/128*PI)
defb 15
Fine della routine di calcolo
I 2 valori in cima allo stack
sono A e 88+80*SIN (A/128*PI)
defb 56
CALL PLOT
23795 CD DC 22    call 8924
Acquisisce il valore di A
23798 F1          pop af
23799 3C          inc a
Controlla se zero (256)
23800 FE 00       cp 0

23802 20 D5       jr nz,LOOP
23804 C9          ret
```

È possibile eseguire operazioni complesse per mezzo di *literal*. Si tratta di chiamate a routine nella ROM che eseguono calcoli complessi sfruttando, in genere, gli ultimi due valori dello stack, ma possono anche essere adottate nella gestione di stringhe. Se, per esempio, i due valori fossero 1234 e 2, la chiamata al Literal 4 da parte della routine CALCULATOR rimuoverebbe questi valori dallo stack, li moltiplicherebbe fra loro ponendo il prodotto nuovamente in cima allo stack, che sarebbe pertanto dimi-

nuito di un valore e conterrebbe alla sommità il numero 2468.
È possibile porre nello stack numeri positivi interi sfruttando

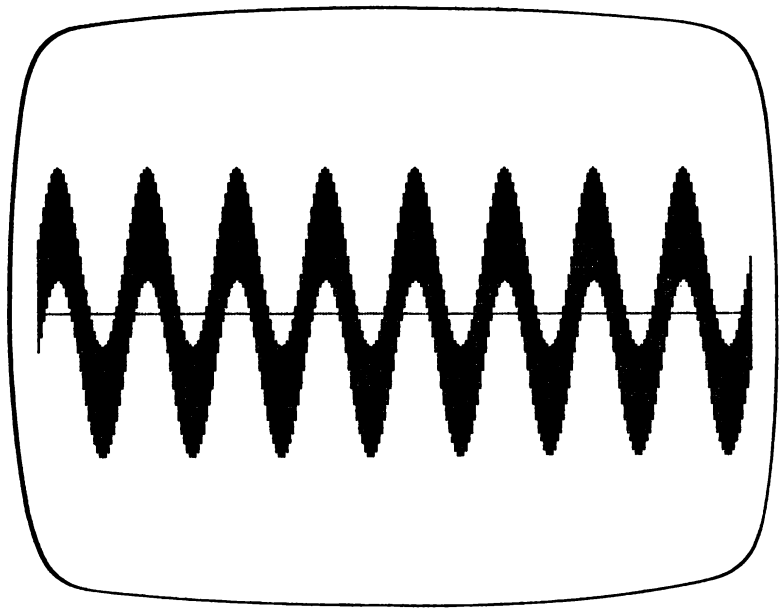
STACK VAL A−CALL 11560 o STACK VAL BC−CALL−11563

Nel programma 2.7 ho sfruttato STACK VAL A per collocare, in ordine, A, 88, 80, A e 128.
La routine USE CALCULATOR viene chiamata dall'istruzione RST 40. La routine CALCULATOR utilizza i byte successivi a RST 40, chiamando ulteriori subroutine come richiesto. Nel nostro programma, il byte successivo è 5, corrispondente al Literal GOSUB DIVIDE. Vengono quindi prelevati gli ultimi due valori dello stack, l'ultimo viene diviso per il precedente e il risultato rimesso sullo stack. Il byte successivo della routine è il Literal 163, che ordina al calcolatore di eseguire STACK PI/2.
Il Literal 49 è DUPLICATE TOP VALUE.
Il Literal 15 è ADD TOP TWO VALUES, e pone la somma sullo stack.
Il Literal 4 è MULTIPLY TOP TWO VALUES.
Il Literal 36 è SINE OF TOP VALUE, che rimpiazza il primo valore dello stack col suo seno.
Esistono molti altri literal che saranno sfruttati in programmi di giochi e discussi nei successivi capitoli. Il Literal 56 è END CALCULATOR ROUTINE CALL che deve essere usato per ritornare al programma in codice macchina. Un altro punto importante da tenere presente durante l'uso della routine CALCULATOR è che, per poter ottenere un sicuro ritorno al BASIC, occorre ripulire lo stack di tutti i valori precedentemente immagazzinati. Alla fine della nostra routine CALCULATOR, abbiamo due valori sullo stack, l'ultimo pari a 88+80*SIN (A/128*P1), e l'altro al valore A. La chiamata all'indirizzo 8924 (PLOT) prende questi due valori, resettando quindi lo stack, e accende l'opportuno pixel. Il valore A viene ora incrementato e il loop si ripete finché A=256 (finché, cioè, A ritorna a 0).
Come potete vedere, i literal della routine CALCULATOR sono molto utili e permettono di trattare in pochi byte anche formule complesse. Sono in effetti necessari più byte per inizializzare lo stack che per eseguire i calcoli.

DRAW

Anche questa routine dispone di due entry point. Il primo (9402) richiede il valore ABS di x nel registro C e il valore di y nel registro B. La coppia di registri DE immagazzina il SGN di x e y: il registro D contiene SGN x (1 se positivo, 255 se negativo) e il registro E, SGN y. Il programma 2.8 è

una versione modificata del programma 2.7 che dimostra l'uso del comando DRAW.



Programma 2.8

```
org 42000 23760

10 PLOT 0,95
20 DRAW 255,0
30 FOR A=0 TO 255
40 PLOT A,120+40*SIN (A/16*PI)
50 DRAW 0,-50
60 NEXT A

Plot 0,95
23760 06 5F        ld b,95
23762 0E 00        ld c,0
23764 CD E5 22     call 8933
Salva H'L'
23767 D9           exx
23768 E5           push hl
23769 D9           exx
Draw 255,0
23770 06 00        ld b,0
23772 0E FF        ld c,255
```

```
23774 16 01        ld d,1
23776 1E 01        ld e,1
23778 CD BA 24     call 9402
Riprende H'L'
23781 D9           exx
23782 E1           pop hl
23783 D9           exx
23784 AF           xor a
L1
23785 F5           push af
23786 CD 28 2D     call 11560
23789 3E 78        ld a,120
23791 CD 28 2D     call 11560
23794 3E 28        ld a,40
23796 CD 28 2D     call 11560
23799 F1           pop af
23800 F5           push af
23801 CD 28 2D     call 11560
23804 3E 10        ld a,16
23806 CD 28 2D     call 11560
23809 EF           rst 40

defb 5 163 49 15 4 31 4 15 56
23819 CD DC 22     call 8924
Salva H'L'
23822 D9           exx
23823 E5           push hl
23824 D9           exx
DRAW 0,-50
23825 06 32        ld b,50
23827 0E 00        ld c,0
23829 16 FF        ld d,255
23831 1E 01        ld e,1
23833 CD BA 24     call 9402
Riprende H'L'
23836 D9           exx
23837 E1           pop hl
23838 D9           exx
23839 F1           pop af
23840 3C           inc a
23841 FE 00        cp 0
23843 20 C4        jr nz,L1
23845 C9           ret
```

Da notare che prima di chiamare la routine DRAW è necessario salvare (in questo caso nello stack) la coppia di registri H'L', che viene sfruttata dalla routine DRAW.
Il valore della coppia di registri H'L' non deve essere modificato dalle routine in codice macchina scritte dall'utente, pena il blocco del sistema

al ritorno al BASIC. La coppia di registri H'L' riassume il valore precedentemente salvato dopo l'esecuzione del comando DRAW. DRAW OVER e INVERSE vengono ottenuti esattamente allo stesso modo di PLOT OVER/INVERSE.
Il secondo entry point per DRAW x,y richiede di porre sullo stack i valori x e y, quest'ultimo in testa, e di eseguire CALL 9335. Desiderando usare questo secondo entry point, sarà bene che studiate il Capitolo 7 per meglio comprendere il modo in cui lo Spectrum immagazzina numeri negativi a cinque byte nello stack di calcolo.

DRAW x,y,a

L'entry point del comando BASIC è 9108. I valori di x,y e a devono essere posti nello stack di calcolo (nell'ordine appena esposto, cioè con a in testa). Sarà ancora utile la lettura del Capitolo 7 per quanto concerne i numeri negativi. Prestate attenzione a non contaminare il valore contenuto nella coppia registri H'L'.

CIRCLE x,y,r

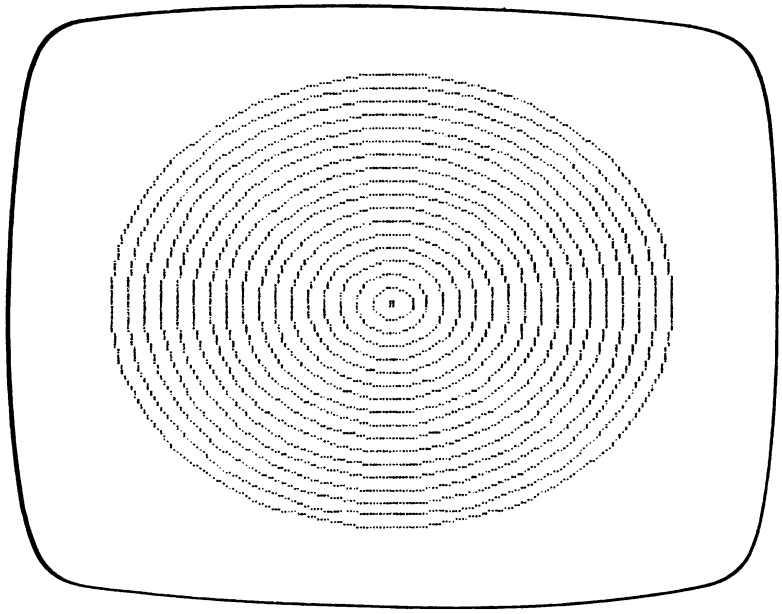
L'entry point di questo comando BASIC è 9005 e richiede di porre nello stack di calcolo i valori x, y e r. Il contenuto della coppia di registri H'L' viene salvato durante la chiamata della routine CIRCLE. Il programma 2.9 mostra come sia possibile disegnare cerchi concentrici per ottenere un effetto tipo sasso nello stagno.

Programma 2.9

```
org 42000 23760

10 REM Circle x,y,r
20 FOR B=1 TO 2
30 FOR A=1 TO 21 STEP 2
40 CIRCLE OVER 1 128,88,A
50 NEXT A
60 NEXT B

For B=1 TO 2

23760 06 02        ld b,2
L2
23762 C5           push bc
```

```
For A=1 TO 21

23763 3E 01        ld a,1
L1
23765 F5           push af


Pone OVER 1

23766 FD 36 57 03 ld (iy+87),3

Stack DATA

23770 3E 80        ld a,128
23772 CD 28 2D     call 11560
23775 3E 58        ld a,88
23777 CD 28 2D     call 11560
23780 F1           pop af
23781 F5           push af
23782 CD 28 2D     call 11560
Salva H'L' (IMPORTANTE !)

23785 D9           exx
23786 E5           push hl
23787 D9           exx

Call CIRCLE


23788 CD 2D 23     call 9005

Riprende H'L'

23791 D9           exx
23792 E1           pop hl
23793 D9           exx
23794 F1           pop af
STEP 2
23795 3C           inc a
23796 3C           inc a
23797 FE 17        cp 23
23799 20 DC        jr nz,L1
23801 C1           pop bc
23802 10 D6        djnz L2

Ripristina OVER 0

23804 FD 36 57 00 ld (iy+87),0
23808 C9           ret
```

Noterete che il programma ha circa gli stessi tempi di esecuzione del BASIC. Ciò è causato dalla relativa lentezza della routine CIRCLE. Desiderando disegnare cerchi, è senz'altro più veloce immagazzinare i punti sotto forma di DATA e sfruttare un loop per visualizzare ogni singolo pixel. Dovendone poi ottenere copie su carta, fate attenzione alla deformazione generata dalla maggior parte delle stampanti grafiche (vedi figura sottostante).



Capitolo 3

I contatori

Gran parte dei programmi di giochi e molti programmi seri richiedono un qualche tipo di contatore. Nei giochi il contatore è usato principalmente per tener conto del punteggio, delle perdite subite, del tempo impiegato... Conosco tre metodi di conteggio e visualizzazione del risultato sullo schermo. Il primo richiede tanta memoria ed è così poco gradevole dal punto di vista estetico che ne darò appena un breve cenno. Questo metodo richiede la predisposizione del massimo numero ammesso sotto forma di DATA. Se per esempio desiderate contare da 0 a 999999, dovrete riservare e inizializzare a 0 sei byte di DATA e sfruttare una routine di stampa di stringa per porre sullo schermo i sei zeri. Ad ogni incremento del contatore occorrerà accertarsi, iniziando dal byte delle unità, di non avere un nove, che richiederebbe la contemporanea manipolazione di due cifre: il nove dovrebbe passare a zero, la cifra alla sua sinistra verrebbe incrementata di uno e così via per tutte le cifre. Dopo la visualizzazione del risultato occorrerebbe anche controllare se per caso si sia raggiunto il massimo numero ammissibile. Il conteggio in su o in giù a passi maggiori di uno richiederebbe un opportuno loop in cui inserire gli incrementi o decrementi unitari.
Il secondo metodo viene usato per contare in su o in giù nel campo da 0 a 65535 ed è illustrato nel programma 3.1.

Programma 3.1

```
org 42000 23760
Azzera il contatore
```

```
23760 01 00 00     ld bc,0
L1
23763 ED 43 B0 5C  ld (23728),bc

Predispone il valore

23767 CD 2B 2D     call 11563

Apre il canale 2

23770 3E 02        ld a,2
23772 CD 01 16     call 5633

Predispone PRINT AT 11,16

23775 3E 16        ld a,22
23777 D7           rst 16

23778 3E 0B        ld a,11
23780 D7           rst 16
23781 3E 10        ld a,16
23783 D7           rst 16

Call PRINT VALUE ON STACK

23784 CD E3 2D     call 11747

Prende il valore e vi somma 1
23787 ED 4B B0 5C  ld bc,(23728)
23791 03           inc bc

Ritorna se zero

23792 78           ld a,b
23793 B1           or c
23794 C8           ret z
E' stato premuto un tasto ?

23795 FD CB 01 6E  bit 5,(iy+1)

23799 28 DA        jr z,L1
23801 C9           ret
```

Questo programma sfrutta una routine che visualizza il valore presente in cima allo stack di calcolo: CALL 11747. Il valore del contatore è immagazzinato in due byte fra le variabili di sistema agli indirizzi 23728/9. In questo caso è necessario prelevare il valore corrente, aggiungervi uno, reimmagazzinare il risultato nel COUNTER, porre nello stack di calcolo il nuovo valore, predisporre i parametri di PRINT AT e chiamare la rou-

tine PRINT VALUE ON STACK. Se il valore contenuto in BC diventa 0 (cioè 65536) si ritorna al BASIC.
Ho inoltre aggiunto dagli indirizzi 23795 a 23800 una routine EXIT, che permette di ritornare al BASIC premendo un tasto. Questa aggiunta è consigliabile per evitare di dover attendere che il loop raggiunga il valore 65535.
La terza forma di conteggio richiede l'uso di CALCULATOR. Fino ad ora abbiamo considerato lo stack di calcolo come un luogo in cui immagazzinare valori senza entrare più in dettaglio. I valori sono immagazzinati nello stack nella forma a cinque byte, cioè ogni numero trattabile dal computer richiede cinque byte di informazione. Maggiori dettagli saranno dati nel Capitolo 7. Nel programma 3.2 sfruttiamo questa caratteristica per contare da 0 in su a passi di 250 senza alcun limite superiore. In effetti, raggiunto il numero 99999999, la routine PRINT STACK VALUE passa a visualizzare i numeri nella loro forma esponenziale (1E+9...), ma senza dubbio cento milioni è un valore sufficientemente alto per qualunque punteggio di giochi.

Programma 3.2

```
org 42000 23760
Usa CALCULATOR

23760 EF            rst 40

LITERAL STACK 0
defb 160

Fine dell' uso di CALCULATOR
defb 56

HL posto a STACKEND-5
dopo l'istruzione defb 56

L1
Copia il valore nel buffer
di stampa

23763 11 00 5B      ld de,23296
23766 01 05 00      ld bc,5

23769 ED B0         ldir

Apre il canale 2

23771 3E 02         ld a,2
23773 CD 01 16      call 5633
```

```
Predispone PRINT AT 11,16
23776 3E 16       ld a,22
23778 D7          rst 16
23779 3E 0B       ld a,11
23781 D7          rst 16
23782 3E 10       ld a,16
23784 D7          rst 16

Call PRINT VALUE ON STACK

23785 CD E3 2D    call 11747

STACK 250


23788 3E FA       ld a,250
23790 CD 28 2D    call 11560

STACK VALUE
23793 21 00 5B    ld hl,23296
23796 ED 5B 65 5C ld de,(23653)
23800 01 05 00    ld bc,5
23803 ED B0       ldir
23805 ED 53 65 5C ld (23653),de

Usa CALCULATOR

23809 EF          rst 40

LITERAL ADD
defb 15
Fine dell'uso di CALCULATOR
defb 56

E' stato premuto un tasto ?


23812 FD CB 01 6E bit 5,(iy+1)
23816 28 C9       jr z,L1

Rimuove l'ultimo valore
presente sullo STACK

23818 2A 65 5C    ld hl,(23653)
23821 2B          dec hl
23822 2B          dec hl
23823 2B          dec hl
23824 2B          dec hl
23825 2B          dec hl
23826 22 65 5C    ld (23653),hl
23829 C9          ret
```

Il programma sfrutta varie importanti routine che meritano alcune spiegazioni. La predisposizione del valore 0 sullo stack di calcolo è ottenuta per mezzo di defb 160, che è il Literal STACK 0; i cinque byte in cima allo stack vengono ora copiati nel buffer di stampa mediante un'istruzione LDIR; il valore di HL è posto dalla routine a (STACKEND−5). Il valore iniziale viene visualizzato in 11,16 dalla routine PRINT VALUE, quindi lo stack viene azzerato.
Consideriamo ora la routine ADD 250. È necessario piazzare sullo stack il valore 250 tramite la routine STACK VAL A e poi il valore già immagazzinato. Ciò si ottiene per mezzo di una istruzione LDIR. Il registro HL punta al valore immagazzinato, il registro DE punta a STACKEND. Dopo l'esecuzione dell'istruzione LDIR occorre mettere il nuovo indirizzo di STACKEND nella variabile di sistema 23653. Il calcolatore viene usato per sommare questi due valori e per ripetere le routine STORE e PRINT VALUE. Viene fornita ancora una routine di uscita dal loop di conteggio, ma notate che lo stack viene ripristinato rimuovendo l'ultimo valore prima del ritorno al BASIC.
Il conteggio all'indietro richiede che venga posto sullo stack il valore di partenza (vedi Capitolo 7 per numeri maggiori di 65535) e venga usato il Literal 3 SUBTRACT. Notate che nella sottrazione il valore in alto viene sottratto da quello sottostante. Alla fine di ogni conteggio sarebbe necessario un controllo per vedere se i byte del valore immagazzinato siano tutti a zero, il che vorrebbe dire fine del conteggio (o se il secondo byte ha il bit 7 a 1, il che rappresenta numeri negativi). Nel conteggio all'indietro è inoltre necessario ripulire lo schermo dal numero precedentemente visualizzato prima di porre il nuovo; infatti, cambiando ad esempio da 1000 a 999, l'ultimo 0 del numero 1000 resterebbe sullo schermo, facendo sembrare così il nuovo numero 9990.
Il programma 3.3 mostra come usare la routine di conteggio come misuratore di riflessi.
Il programma BASIC è autoesplicante ed è in gran parte costituito dalla stampa delle istruzioni di gioco. Vengono usati comandi DATA per immagazzinare i parametri di x, y, h, w e a$ per la visualizzazione espansa presentata nel Capitolo 2.


PROVARIFLESSI
Questo programma dimostra il conteggio in codice macchina presentando un PROVARIFLESSI
Lo Spectrum sceglie una lettera (da A a Z) e la visualizza in PAPER 6 e INK 6.
Il codice macchina cambia INK in INK 0, seleziona CAPS LOCK e misura il tempo impiegato a premere il tasto corrispondente.
La lettera selezionata apparira' dopo un ritardo casuale.
PREMI UN TASTO PER GIOCARE


PROVARIFLESSI
CONTEGGIO
60
Z
ECCELLENTE
DI NUOVO ? s/n

Programma 3.3

```
 12>PAPER 6:CLS
 15 DATA 55,143,1,2,"CONTEGGIO"
 20 DATA 88,63,1,2,"LENTO"
 30 DATA 65,63,1,2,"DISCRETO"
 40 DATA 90,63,1,2,"BUONO"
 50 DATA 45,63,1,2,"MOLTO BUONO"
 60 DATA 53,63,1,2,"ECCELLENTE"
 80 DATA 35,63,1,2,"ADDORMENTATO"
 90 DATA 10,31,1,2,"DI NUOVO ? s/n"
 95 DATA 70,125,5,2,"GRAZIE"
 96 DATA 110,79,3,2,"e"
 97 DATA 33,50,5,2,"ARRIVEDERCI"
100 DATA 10,79,5,2,"FERMA IL NASTRO"
105 DATA 25,167,1,2,"PROVARIFLESSI"
110 DATA 15,164,2,2,"_______________"
120 DATA 20,15,1,2,"PREMI UN TASTO"
125 RESTORE 100
130 FOR a=1 TO 4
140 PAUSE 25
160 INK 2: GO SUB 9000
170 NEXT a
180 PAUSE 0: BORDER 5: PAPER 6: INK 0: CLS
190 RESTORE 105
220 INK 0: GO SUB 9000
```

```
 250 PRINT AT 3,0;"Questo  programma  dimostr
a  il  conteggio  in  codice  macchina  present
ando un PROVARIFLESSI"
 260 PRINT '"Lo Spectrum  sceglie una lettera
(da A a Z ) e  la  visualizza inPAPER 6 e INK
 6."
 270 PRINT '"Il  codice  macchina  cambia INK
in INK 0, seleziona CAPS LOCK  emisura  il  t
empo   impiegato  apremere il tasto corrispon
dente."
 280 PRINT '"La lettera selezionata apparira'
dopo un ritardo casuale."
 290 PRINT AT 20,3;"PREMI UN TASTO PER GIOCAR
E"
 300 IF INKEY$<>"" THEN  GO TO 300
 302 IF INKEY$="" THEN  GO TO 302
 305 CLS
 310 RESTORE 105: INK 1: GO SUB 9000
 320 INK 5: GO SUB 9000
 330 RESTORE 15: INK 2: GO SUB 9000
 350 INK 0: PLOT 102,114: DRAW 50,0: DRAW 0,-
38: DRAW -50,0: DRAW 0,38
 380 LET z=INT (RND*26+65)
 390 LET x=111: LET y=111: LET h=4: LET w=4:
LET a$=CHR$ z
 400 INK 6: LET c=USR 32393
 410 FOR a=1 TO RND*200+50: NEXT a
 420 INK 0
 430 LET c=USR 23760
 440 LET a=PEEK 23728+256*PEEK 23729
 450 IF a>=140 OR a=0 THEN  RESTORE 80
 460 IF a<140 THEN  RESTORE 20
 470 IF a<120 THEN  RESTORE 30
 480 IF a<100 THEN  RESTORE 40
 490 IF a<85 THEN  RESTORE 50
 500 IF a<70 THEN  RESTORE 60
 510 GO SUB 9000
7000 RESTORE 90: INK 1: GO SUB 9000
7010 IF INKEY$<>"" THEN  GO TO 7010
7020 IF INKEY$="" THEN  GO TO 7020
7030 LET d$=INKEY$
7040 IF d$<>"s" THEN  GO TO 8000
7050 LET c=USR 23842: GO TO 350
8000 RESTORE 95
8005 POKE 23843,20
8010 LET c=USR 23842
8015 POKE 23843,18
8030 FOR a=1 TO 3
8040 GO SUB 9000
8050 NEXT a
```

```
8060 STOP
9000 READ x,y,h,w,a$
9010 LET c=USR 32393
9020 RETURN
9800 CLEAR 32334: LOAD ""CODE : GO TO 5


org 42000 23760
Trova la variabile z

23760 3E 7A       ld a,122
Immagazzina in 23681
23762 01 81 5C    ld bc,23681
23765 CD 7D 7E    call 32381

Predispone CAPS LOCK
23768 FD CB 30 DE set 3,(iy+48)

Scambia l'attributo di colore

23772 21 00 58    ld hl,22528
L1
23775 7E          ld a,(hl)
23776 FE 36       cp 54
PAPER 6 INK 6
23778 20 02       jr nz,L2
23780 36 30       ld (hl),48

Pone PAPER 6 INK 0
L2
23782 23          inc hl
23783 7C          ld a,h
23784 FE 5B       cp 91
Fine di ATTRIBUTES ?
23786 20 F3       jr nz,L1

Inizia il conteggio

23788 01 00 00    ld bc,0
Pone il conteggio sullo stack
L3
23791 C5          push bc
Pone il conteggio su CALCULATOR
23792 CD 2B 2D    call 11563

Predispone PRINT AT 6,14
23795 3E 02       ld a,2
23797 CD 01 16    call 5633
23800 3E 16       ld a,22
```

```
23802 D7          rst 16
23803 3E 06       ld a,6
23805 D7          rst 16
23806 3E 0E       ld a,14
23808 D7          rst 16

Visualizza il conteggio
23809 CD E3 2D    call 11747

Prende il conteggio e
vi somma 1
23812 C1          pop bc
23813 03          inc bc

Controllo conteggio < 65535
23814 78          ld a,b
23815 B1          or c
23816 28 0F       jr z,END

E' stato premuto un tasto ?


23818 FD CB 01 6E bit 5,(iy+1)
23822 28 DF       jr z,L3

Trova INKEY$

23824 3A 08 5C    ld a,(23560)
Confronta con a$
23827 21 81 5C    ld hl,23681
23830 BE          cp (hl)
23831 20 D6       jr nz,L3
END
Immagazzina il conteggio
23833 ED 43 B0 5C ld (23728),bc
Ritorna ai caratteri minuscoli
23837 FD CB 30 9E res 3,(iy+48)
23841 C9          ret

Cancella le 18 linee inferiori

23842 06 12       ld b,18
23844 CD 44 0E    call 3652

23847 C9          ret
```

Capitolo

I numeri casuali 4

I numeri casuali sono fondamentali in molti programmi di gioco, come il ritardo casuale e la scelta casuale della lettera nel programma 3.3 PROVARIFLESSI del Capitolo 3. È estremamente facile ottenere un numero casuale sfruttando il codice macchina; in funzione del suo campo di variazione sono disponibili due metodi.
Il primo metodo, che non produce un numero veramente casuale ma sufficientemente valido per molti giochi, sfrutta le variabili FRAMES e SEED in maniera estremamente singolare. Questo metodo potrebbe esservi utile se vi serve un numero casuale che vari nel campo 0–1, 0–3, 0–7, 0–15, 0–31, 0–63, 0–127, o 0–255. La routine richiede di porre nella coppia di registri HL il valore contenuto nella variabile SEED (23670), nella coppia di registri DE il byte di ordine alto di SEED e il byte d'ordine basso di FRAMES (23672); vengono sommati i valori contenuti nelle due coppie di registri ed il risultato immagazzinato nella variabile SEED per il successivo numero casuale. Il valore di L viene posto nel registro A e si mascherano i bit indesiderati tramite un AND a; tramite l'opportuna scelta della maschera si può scegliere il campo di variazione del numero casuale. Il programma 4.1 fornisce INT(RND*32), mascherando i tre bit di ordine alto dell'accumulatore, in modo da restringere il campo di variazione del numero casuale da 0 a 31.

Programma 4.1

```
org 42000 23760
10 FOR a=1 TO 10
```

```
20 PRINT INT (RND*32)
30 NEXT a

10 numeri casuali
23760 06 0A        ld b,10
L1
23762 C5           push bc
Load HL,(SEED)
23763 2A 76 5C     ld hl,(23670)
23766 ED 5B 77 5C  ld de,(23671)
23770 19           add hl,de

SEED modificato per il numero
successivo
23771 22 76 5C     ld (23670),hl

Maschera i BIT 7-5
23774 7D           ld a,l

23775 E6 1F        and 31

Salva il numero casuale
23777 F5           push af

Predispone PRINT AT b,10
23778 3E 02        ld a,2
23780 CD 01 16     call 5633
23783 3E 16        ld a,22
23785 D7           rst 16
23786 78           ld a,b
23787 D7           rst 16
23788 3E 0A        ld a,10
23790 D7           rst 16

Prende il numero casuale e lo
pone sullo STACK di CALCULATOR
23791 F1           pop af
23792 CD 28 2D     call 11560
23795 CD E3 2D     call 11747
Lo visualizza


Riprende il contatore di numeri
23798 C1           pop bc

Ripete per 10 numeri
23799 10 D9        djnz L1
23801 C9           ret
```

```
NUMERI CASUALI GENERATI
DAL PROGRAMMA 4.1

6
22
23
8
9
25
24
5
1
10
```

Come già accennato il programma potrà esservi utile per fornire numeri quasi casuali compresi in uno dei campi di variazione presentati. È possibile migliorare il programma in modo da ottenere altri estremi dei campi di variazione, ad esempio, INT(RND*23) tramite l'aggiunta di:

```
LD A, H
AND 15
LD H, A
LD A, L
AND 7
ADD A, H
```

Avrete notato che nel programma 4.1 il registro B viene usato come contatore di loop (dieci volte) e che questo valore viene anche usato nella routine di PRINT AT b, 10. L'esempio di stampa vi fornisce un'idea circa la casualità dei numeri generati dalla routine.
Per produrre un numero casuale in modo analogo al comando BASIC INT(RND*n), è necessario usare la routine RND, purtroppo non disponibile tramite un semplice CALL RND, non essendo dotata di istruzione di ritorno. La routine si trova nella ROM da 9725 a 9765 e sfrutta il calcolatore per modificare il valore di SEED. Essa preleva il valore da SEED, lo tratta e termina col nuovo valore duplicato in cima allo stack. Il valore in cima viene prelevato e usato per aggiornare SEED. Il valore rimanente viene successivamente modificato per fornire un valore compreso fra 0 e 1 (sempre minore di uno) che è usato per RND. Per sfruttare questa routine è necessario copiarla nella RAM e chiamarla dal vostro programma come subroutine. Il programma 4.2 mostra come usare questo metodo per ottenere INT(RND*12345) e stampa 44 valori usando CHR$ CODE 6 per ottenere la stampa in due colonne.

Programma 4.2

```
org 42000 23760

10 INT (RND*12345)

CLS

23760 3E 02        ld a,2
23762 CD 01 16     call 5633
23765 CD 6B 0D     call 3435

44 numeri casuali

23768 06 2C        ld b,44
L1
23770 C5           push bc

COPY della ROM da 9725 a 9765

23771 ED 4B 76 5C ld bc,(23670)
23775 CD 2B 2D     call 11563

23778 EF           rst 40
STACK (1)
defb 161
+
defb 15
STACK (75)
defb 52 55 22
*
defb 4
STACK (65537)
defb 52 128 65 0 0 128
LITERAL
defb 50 2
STACK (1)
defb 161
-
defb 3
Duplicate
defb 49
Fine di CALC ROUTINE
defb 56

Pone nei REGISTRI BC la cima
dello stack (INT 0-65535)
23797 CD A2 2D     call 11682
Salva il nuovo SEED
23800 ED 43 76 5C ld (23670),bc
```

```
Manipola l'EXPONENT BYTE
23804 7E          ld a,(hl)
23805 A7          and a
23806 28 03       jr z,L2
23808 D6 10       sub 16
23810 77          ld (hl),a
L2
STACK 12345
23811 01 39 30    ld bc,12345
23814 CD 2B 2D    call 11563
23817 EF          rst 40
*
defb 4
LITERAL 'INTEGER'
defb 39
Fine di CALC ROUTINE

defb 56
Visualizza il valore su STACK
23821 3E 02       ld a,2
23823 CD 01 16    call 5633
23826 CD E3 2D    call 11747
PRINT ,
23829 3E 06       ld a,6
23831 D7          rst 16

Controlla il contatore

23832 C1          pop bc
23833 10 BF       djnz L1
23835 C9          ret
```

NUMERI CASUALI GENERATI
DAL PROGRAMMA 4.2

1857	3558
7666	7091
1037	3758
10296	12044
3058	7178
7727	11711
3057	7150
2940	10693

La routine RND usa vari literal che possono essere utili anche in altre routine:

Literal 161 è STACK NUMBER 1
Literal 52 è STACK DATA (tipicamente un numero in forma compressa, vedi Capitolo 7)
Literal 50 è n-mod-m
Literal 2 è DELETE (il quoziente)

Gli ultimi due literal trovano probabilmente ben poche applicazioni nei programmi di giochi. La routine –CALL 11682– viene usata dalla ROM per prelevare il valore in cima allo stack e porre il valore intero nella coppia di registri BC. Il valore intero è in questo caso arrotondato per eccesso o per difetto al più vicino numero intero. Esiste anche una routine simile –CALL 11733– che piazza la cima dello stack in modo simile nel registro A. La routine copiata finisce in 23810.
La routine da 23811 a 23820 moltiplica ulteriormente il valore RND sullo stack. Il valore 12345 viene immagazzinato nello stack di calcolo e moltiplicato per RND. Viene sfruttato un nuovo literal, il Literal 39 che rimuove la parte decimale dal risultato di 12345*RND arrotondando all'intero inferiore (come nel comando BASIC RND), lasciando quindi INT (RND*12345) in cima allo stack di calcolo. Viene poi usata la routine PRINT VALUE per visualizzare i 44 valori ottenuti in due colonne; come potete vedere, i risultati sono veramente casuali.
Il programma dimostrativo 4.3 mostra l'uso di RND. In questo caso RND viene sfruttato per ottenere grattacieli di altezza casuale in caratteri e colori casuali, che potrebbero essere adottati in programmi tipo "attacco aereo". Tutti gli UDG per ottenere un programma completo sono già predisposti e potreste quindi sviluppare il programma. Il programma RNDDEMO usa il BASIC per ottenere la prima visualizzazione della durata di pochi secondi che viene sostituita dalle visualizzazioni generate dalla routine in codice macchina, in modo da dimostrarne la velocità e la funzione RND.
Desiderando usare l'istruzione BASIC RND con chiamate a routine in codice macchina, sarà importante eseguire queste tramite LET *variabile* = USR *indirizzo*, evitando di usare RANDOMIZE USR *indirizzo* se non vi importa la visualizzazione, al ritorno al BASIC, del valore contenuto nella coppia di registri BC. Nella routine in codice macchina i colori PAPER e INK vengono predisposti usando LD (IY+83), 47 invece di LD A,47: LD (IY+83) A, come negli esempi precedenti, risparmiando così memoria e mostrando l'utilità del registro IY come puntatore. Questo è ulteriormente dimostrato nel programma in cui IY viene inizializzato a 23296, indirizzo iniziale del buffer di stampa, e usato per LD (23296), 22 (AT) e LD (23299), 16 (INK). Quando cambiate il valore contenuto in IY, comunque,

abbiate cura di ripristinarlo a 23610 prima di eseguire chiamate a routine ROM che lo sfruttino; potreste, in caso contrario, fare impazzire il sistema.
Prima di chiamare la routine di generazione di numeri casuali, il registro A viene caricato col numero RND*n, che viene immagazzinato all'indirizzo 23681 (indirizzo non sfruttato fra le variabili di sistema). Questo numero viene poi prelevato dalla routine RND e usato per ottenere INT(RND*n); il valore così ottenuto viene posto nel registro A per mezzo di CALL 11733 prima del ritorno dalla routine. Il buffer di stampa viene usato per immagazzinare ordinatamente le informazioni di PRINT:AT(22), x, y, INK (16), b$, a$.

Programma 4.3

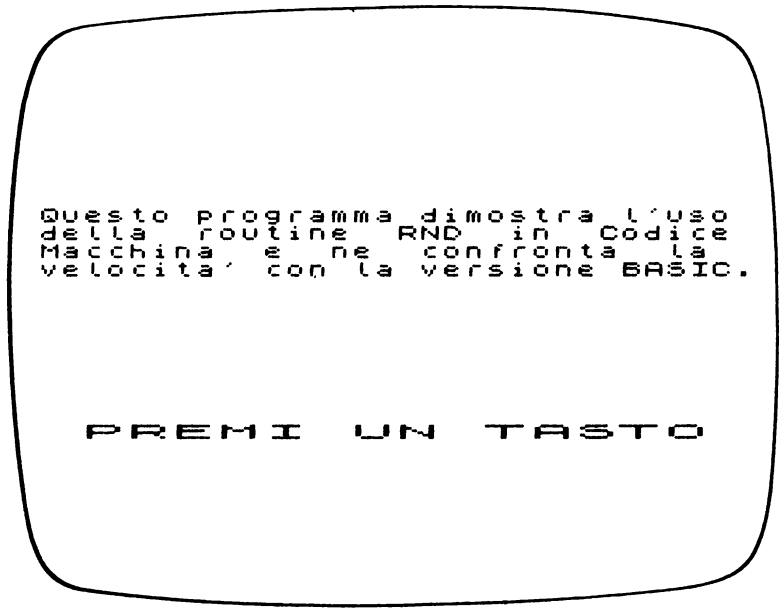
```
  10 BORDER 0: PAPER 5: INK 7: C
LS
  20 FOR c=31 TO 0 STEP -1
  30 LET l=19-INT (RND*10)
  40 LET a$=CHR$ INT (RND*2+148)
  45 LET b$=CHR$ INT (RND*4)
  50 PRINT AT l-1,c;"▟▙"
  60 FOR a=l TO 20
  70 PRINT AT a,c;CHR$ 16+b$+a$
  80 NEXT a
  90 PRINT AT a,c;"▀▀"
 100 NEXT c
 110 LET x=0: LET y=0: LET a$="
▶▶"
 130 PRINT AT x,y;a$
 140 PRINT #0;"▀▀ FUEL▌10000 ▀▀ BOMBS▌
100 ▀▀ SHOTS▌ 100 ▀▀"
 150 RETURN
 500 DATA 10,79,5,2,"FERMA IL NA
STRO"
 510 DATA 15,167,1,2,"NUMERI CAS
UALI"
 520 DATA 15,164,1,2,"_____
_____"
 525 DATA 15,15,1,2,"PREMI UN TA
STO"
 530 DATA 47,164,2,4,"BASIC"
 540 DATA 10,164,3,2,"CODICE MAC
CHINA"
 600 BORDER 4: PAPER 6: INK 0: C
LS
 605 RESTORE 500
 610 FOR a=1 TO 4
 620 INK a: GO SUB 6000
 630 PAUSE 50: NEXT a
 640 PAUSE 0: CLS
 645 PRINT INK 0;AT 8,0;"Questo
programma dimostra l'uso della
routine RND in Codice Macchin
a e ne confronta la velocit
a con la versione BASIC."
 650 RESTORE 525
 655 PAUSE 100
 660 FLASH 1: INK 2: PAPER 7: GO
SUB 6000: FLASH 0
 670 PAUSE 0: CLS
1000 LET z=1
1010 GO SUB 2000
1020 GO SUB 10
1025 LET z=z+1
```

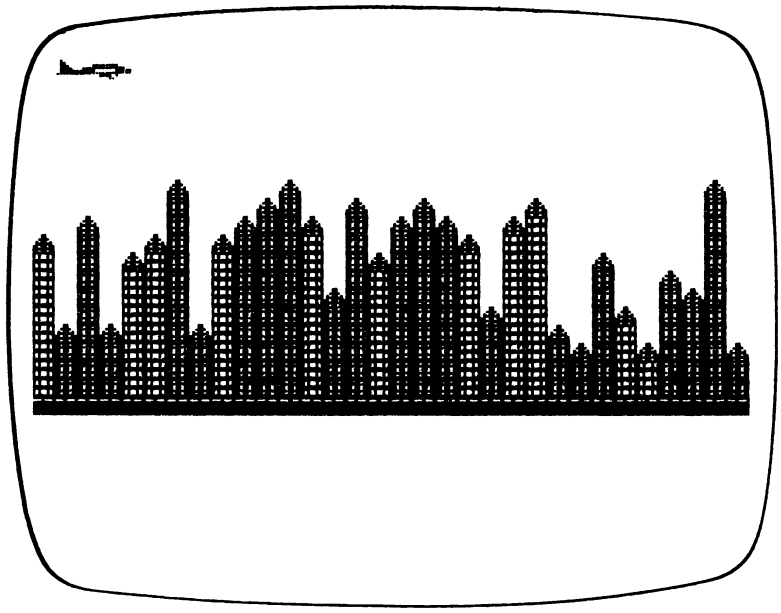
```
1030 GO SUB 2000
1040 GO SUB 3000
1045 LET z=z+1
1050 GO TO 1010
2000 FOR a=1 TO 400: NEXT a: CLS
2010>PAPER 8: INK 0: RESTORE 530
+(10*(z/2=INT (z/2))): GO SUB 80
00
2020 FOR a=1 TO 200: NEXT a: RET
URN
3000 LET c=USR 23760: GO TO 110
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 32334: LOAD "udg"CODE
 USR "a": LOAD ""CODE : GO TO 50
0
```

```
org 42000 23760

BORDER 0

23760 3E 00       ld a,0
23762 CD 9B 22    call 8859

PAPER 5 INK 7

23765 FD 36 53 2F ld (iy+83),47

CLS

23769 3E 02       ld a,2
23771 CD 01 16    call 5633
23774 CD 6B 0D    call 3435

Usa IY come PUNTATORE

23777 FD 21 00 5B ld iy,23296

23781 FD 36 00 16 ld (iy+0),22
23785 FD 36 03 10 ld (iy+3),16

Resetta IY
23789 FD 21 3A 5C ld iy,23610

Conta le colonne

23793 3E 1F       ld a,31
L1
23795 32 02 5B    ld (23298),a
```

```
19-INT (RND*10)

23798 3E 0A       ld a,10
23800 32 81 5C    ld (23681),a
23803 CD 61 5D    call RND
23806 6F          ld l,a
23807 3E 13       ld a,19
23809 95          sub l
23810 32 01 5B    ld (23297),a

INT (RND*2+148)

23813 3E 02       ld a,2
23815 32 81 5C    ld (23681),a
23818 CD 61 5D    call RND
23821 C6 94       add a,148
23823 32 05 5B    ld (23301),a

INT (RND*4)

23826 3E 04       ld a,4
23828 32 81 5C    ld (23681),a
23831 CD 61 5D    call RND
23834 32 04 5B    ld (23300),a

PRINT AT x-1,y CHR$ 147

23837 3E 02       ld a,2
23839 CD 01 16    call 5633
23842 3E 16       ld a,22

23844 D7          rst 16
23845 3A 01 5B    ld a,(23297)
23848 3D          dec a
23849 D7          rst 16
23850 3A 02 5B    ld a,(23298)
23853 D7          rst 16
23854 3E 93       ld a,147
23856 D7          rst 16
L2

FOR a=x TO 20
PRINT AT a,y CHR$ 16+b$+a$
23857 11 00 5B    ld de,23296
23860 01 06 00    ld bc,6
23863 CD 3C 20    call 8252
23866 3A 01 5B    ld a,(23297)
23869 3C          inc a
23870 32 01 5B    ld (23297),a
23873 FE 15       cp 21
```

```
NEXT a
23875 20 EC        jr nz,L2

PRINT AT a,y CHR$ 153

23877 3E 99        ld a,153
23879 32 05 5B     ld (23301),a
23882 3E 07        ld a,7
23884 32 04 5B     ld (23300),a
23887 11 00 5B     ld de,23296
23890 01 06 00     ld bc,6
23893 CD 3C 20     call 8252
23896 3A 02 5B     ld a,(23298)
23899 3D           dec a
23900 FE FF        cp 255

COLONNA SUCCESSIVA

23902 20 93        jr nz,L1
23904 C9           ret

ROUTINE RND

RND
23905 ED 4B 76 5C ld bc,(23670)
23909 CD 2B 2D     call 11563
23912 EF           rst 40
defb 161 15 52 55 22 4 52 128
defb 65 0 0 128 50 2 161 3 49 56
23931 CD A2 2D     call 11682
23934 ED 43 76 5C ld (23670),bc
23938 7E           ld a,(hl)
23939 A7           and a
23940 28 03        jr z,L3
23942 D6 10        sub 16
23944 77           ld (hl),a

INT (RND*PEEK (23681))

L3
23945 3A 81 5C     ld a,(23681)
23948 CD 28 2D     call 11560
23951 EF           rst 40
defb 4 39 56

VALORE in A

23955 CD D5 2D     call 11733
23958 C9           ret
```


Questo programma dimostra l'uso
della routine RND in Codice
Macchina e ne confronta la
velocita' con la versione BASIC.
PREMI UN TASTO



Capitolo

La tastiera 5

Esistono due metodi di lettura della tastiera in codice macchina. Entrambi sono efficaci e la scelta dell'uno o dell'altro dipende esclusivamente dal tipo di programma.

Uso della variabile LAST KEY

La ROM dello Spectrum contiene una routine KEYSCAN e DECODE, eseguita automaticamente 50 volte al secondo. La routine, oltre a scandire la tastiera e decodificare il tasto premuto, aggiorna il contatore FRAMES.
Il codice del carattere corrispondente all'ultimo tasto premuto viene immagazzinato nella variabile LAST KEY (23560) e, quando è stato premuto un tasto, il bit 5 della locazione (IY+1) viene posto a uno. Incidentalmente, l'istruzione HALT attende l'esecuzione della successiva routine KEYSCAN prima di proseguire l'esecuzione del programma in codice macchina. Conoscendo questi dati, possiamo scrivere un programma che contenga l'equivalente di PAUSE 0.

Programma 5.1

```
org 42000 23760

ATTENDE CHE VENGA PREMUTO
UN TASTO
```

```
L1
23760 FD CB 01 6E bit 5,(iy+1)
23764 28 FA       jr z,L1
23766 FD CB 01 AE res 5,(iy+1)

CONTINUA L'ESECUZIONE DEL
PROGRAMMA

23770 C9          ret
```

Come potete vedere nel programma 5.1, il bit 5 della locazione (IY+1) viene continuamente controllato per vedere se è settato a uno e il programma continuerà quando e solo quando ciò è stato verificato (il che vuol dire che è stato premuto un tasto). Notare che è necessario resettare il bit 5 della cella (IY+1) dopo averlo trovato a uno, altrimenti l'eventuale esecuzione di una simile routine di PAUSE prima di un successivo KEYSCAN troverebbe il bit 5 ancora a uno e non effettuerebbe quindi alcuna attesa.

Uso dell'istruzione IN A, (C)

Questa istruzione è identica al comando BASIC IN e legge la mezza riga identificata dal valore contenuto nel registro BC, come spiegato nel manuale dello Spectrum. Il valore restituito in A indica quali tasti siano stati premuti in quella semiriga. Ogni semiriga contiene cinque tasti e i bit 4–0 dell'accumulatore contengono il relativo stato (i bit 7–5 sono normalmente a uno); il bit 0 viene usato per indicare lo stato del tasto più esterno della semiriga, mentre il bit 4 riporta lo stato del tasto più interno. Quando nessun tasto è premuto tutti i bit sono posti a uno ed A contiene 255. La pressione di un tasto causa il reset del bit corrispondente. Premendo, ad esempio, il tasto 0, si resetta il bit 0 ed A contiene quindi il valore 254. Come potete vedere, con questo metodo siamo in grado di riconoscere la pressione contemporanea di più di un tasto, controllando o il valore di A o i suoi singoli bit. Questo accorgimento si rivela utile nei giochi per due o più giocatori o per muovere oggetti in due direzioni (ad esempio in su e in avanti) contemporaneamente. Il programma 5.2 dimostra l'uso del comando IN per attendere la pressione del tasto 0.

Programma 5.2

```
org 42000 23760

LEGGE LA MEZZA RIGA 6-0

L1
23760 01 FE EF     ld bc,61438
23763 ED 78        in a,(c)

CONTROLLA SE E' STATO PREMUTO
IL TASTO 0

23765 CB 47        bit 0,a
23767 C8           ret z
23768 18 F6        jr L1
```

La routine KEYSCAN viene chiamata 50 volte al secondo, anche durante l'esecuzione del nostro programma in codice macchina, che viene così rallentato. Desiderando maggiore velocità di esecuzione, ad esempio durante l'esecuzione di PIXELSCROLLING, possiamo evitare questa interruzione sfruttando l'istruzione DI. Ovviamente, ad interrupt disabilitati non possiamo usare la routine LAST KEY per ottenere dati dalla tastiera (salvo la possibilità di chiamare direttamente la routine nella ROM) e dobbiamo ricorrere all'istruzione IN. Il programma 5.3 dimostra l'uso dell'istruzione DI.

Programma 5.3

```
org 42000 23760

DISABILITA

23760 F3           di

LEGGE LA MEZZA RIGA 6-0

L1
23761 01 FE EF     ld bc,61438
23764 ED 78        in a,(c)

CONTROLLA SE E' STATO PREMUTO
IL TASTO 0

23766 CB 47        bit 0,a
23768 20 F7        jr nz,L1

ABILITA

23770 FB           ei
23771 C9           ret
```

È importante ricordarsi di inserire un'istruzione EI prima di ritornare al BASIC; senza questo accorgimento la tastiera continuerà a non funzionare. Provate il programma senza inserire l'istruzione EI e vedete cosa succede. Non avverrà nessun danno permanente, ma sarà necessario spegnere e quindi riaccendere lo Spectrum per riottenere il funzionamento della tastiera.
Il programma 5.4 mostra come ottenere PAUSE n per generare ritardi o, come in BASIC, continuare se viene premuto un tasto.
Il programma sfrutta il fatto che la scansione della tastiera avviene ogni 0.02 secondi e che l'istruzione HALT attende una scansione di tastiera.

Programma 5.4

```
org 42000 23760

PREDISPONE UN RITARDO DI
10 SECONDI

23760 01 F4 01     ld bc,500
L1
LOOP DI ATTESA
23763 76           halt

E'STATO PREMUTO UN TASTO ?

23764 FD CB 01 6E bit 5,(iy+1)
23768 20 05        jr nz,L2
DECREMENTA IL CONTATORE
23770 0B           dec bc
23771 78           ld a,b
23772 B1           or c
23773 20 F4        jr nz,L1
L2

23775 FD CB 01 AE res 5,(iy+1)
23779 C9           ret
```

Il programma 5.5 sfrutta il metodo della variabile LAST KEY per leggere dati da tastiera e ottenere un semplice programma tipo macchina da scrivere, che permette cancellazioni e modifiche di quanto digitato.
La posizione corrente sullo schermo è indicata da un cursore lampeggiante. Da notare che la pressione di CAPS SHIFT e 2 sposta la posizione del cursore alla successiva PRINT COMMA TAB; CAPS SHIFT e 5 genera il ritorno indietro di uno spazio (da notare l'errore contenuto nella ROM che non permette di tornare dall'inizio della linea 1 alla fine della linea

0); ENTER genera una nuova linea. Lavorando in codice macchina il tasto BREAK è inutilizzabile e ho quindi sfruttato STOP per ottenere il ritorno al BASIC.
Come con l'istruzione BASIC INKEY$, non vengono letti né i tasti doppiamente shiftati, né quelli di modo grafico. Desiderando controllare, o visualizzare, i tasti in modo E doppiamente shiftati, avrete bisogno di costruire una tabella DATA di corrispondenza fra caratteri non shiftati e i loro equivalenti doppiamente shiftati. Quindi, se l'istruzione LD A, (LAST KEY) restituisce nell'accumulatore il valore 14, indicando la pressione di entrambi i tasti di shift sarà necessario eseguire una routine che legga il successivo tasto premuto e converta il suo valore non shiftato nell'equivalente doppiamente shiftato, sfruttando la tabella DATA.

Programma 5.5

```
20>DATA 10,79,5,2,"FERMA IL NASTRO"
30 DATA 45,170,4,1,"MACCHINA PER SCRIVERE"
40 DATA 7,150,2,3,"__________"
50 DATA 20,15,1,2,"PREMI UN TASTO"
60 DATA 39,170,1,2,"ISTRUZIONI"
65 RESTORE
70 BORDER 4: PAPER 6: CLS
80 FOR A=1 TO 4
```

```
  85 PAUSE 20
  90 INK A: PAPER 8: GO SUB 8000
 100 NEXT A
 110 PAUSE 0: PAPER 7: CLS
 120 INK 2: PAPER 8: GO SUB 8000
 130 INK 0: PRINT AT 3,0;"Questo programma di
mostra come  acquisire dati dalla tastiera
sfruttando il codice macchina,  la variabile
LAST KEY e          controllando il BIT 5 diFL
AGS   per vedere se sia stato premuto un tast
o."
 140 PRINT '"E' possibile modificare quanto
scritto spostando il cursore    tramite il ta
sto 5."
 150 PRINT "ENTER genera un ""a capo"", BREAK
 uno SPAZIO  ; 'STOP' restituisceil controllo
 al BASIC."
 160 PRINT "Il codice macchina e' lungo solo4
3 byte"
 200 RESTORE 50: INK 2: GO SUB 8000
 210 PAUSE 0: BRIGHT 1: CLS
 220 INK 0: RANDOMIZE USR 23760
 230 STOP
8000 READ x,y,h,w,a$
8010 RANDOMIZE USR 32393
8020 RETURN
9990 CLEAR 32334: LOAD ""CODE : GO TO 20



org 42000 23760
KEYDEMO


Apre il canale 2

23760 3E 02       ld a,2
23762 CD 01 16    call 5633

PONE OVER 1

L1
23765 FD 36 57 03 ld (iy+87),3

CURSORE LAMPEGGIANTE DURANTE
IL LOOP DI ATTESA

23769 3E 8F       ld a,143
23771 D7          rst 16
23772 3E 08       ld a,8
```

```
23774 D7          rst 16
23775 3E 8F       ld a,143
23777 D7          rst 16
23778 3E 08       ld a,8
23780 D7          rst 16

ATTENDE CHE VENGA PREMUTO
UN TASTO

23781 FD CB 01 6E bit 5,(iy+1)
23785 28 EA       jr z,L1
23787 FD CB 01 AE res 5,(iy+1)

PONE OVER 0

23791 FD 36 57 00 ld (iy+87),0

LEGGE LAST KEY

23795 3A 08 5C    ld a,(23560)


CONTROLLA SE E' 'STOP'

23798 FE E2       cp 226
23800 C8          ret z

PRINT CHR$

23801 D7          rst 16
23802 18 D9       jr L1
```

Capitolo

Animazione 6

Ora che siamo in grado di acquisire dati dalla tastiera e di visualizzare qualunque cosa, possiamo esaminare alcuni metodi di animazione, parte essenziale di ogni programma di giochi in tempo reale. In questo capitolo tratteremo diversi metodi in grado di dare l'impressione del movimento:

1. Muovendo gli oggetti visualizzati — navi spaziali, invasori, gente, ...
2. Muovendo lo sfondo — stelle, pianeti, grattacieli, ...
3. Muovendo oggetti e sfondo contemporaneamente.

Il tipo di movimento usato dalla gran parte dei programmatori è a passi di un carattere; questo permette giochi molto veloci e l'uso di tutti i colori. Lo schermo è diviso in 24 righe di 32 colonne e, quando un carattere viene spostato, si muove dalla sua posizione attuale alla riga/colonna adiacente; questo movimento è abbastanza grossolano, ma non eccessivamente brusco. Sfruttando il movimento a pixel, specialmente sugli oggetti visualizzati, è necessario ridisegnare i caratteri. Se conoscete l'uso del comando BASIC PLOT, capirete che questo ci limita all'uso di due soli colori (PAPER e INK).

Movimento degli oggetti visualizzati a passi di un carattere

Occorre innanzitutto definire la posizione iniziale del carattere, quindi, usando uno dei metodi presentati nel Capitolo 5, acquisire dati dalla ta-

stiera aggiornando conseguentemente la posizione del carattere. L'aggiornamento della posizione del carattere viene generalmente eseguito su un set duplicato di dati di posizione iniziale. Quando l'aggiornamento viene completato, abbiamo due set di dati: il primo contiene l'attuale posizione di schermo ed il secondo la nuova. Viene eseguito ogni controllo, per esempio per vedere se la nuova posizione è ancora sullo schermo, ed i nuovi dati corretti di conseguenza. Per dare l'impressione di movimento privo di sfarfallio, rimuoviamo il carattere esistente e lo rivisualizziamo immediatamente nella sua nuova posizione. In genere io uso OVER 1 e rivisualizzo il carattere nella vecchia posizione, quindi lo rimuovo e, usando ancora OVER 1, lo visualizzo nella nuova posizione. Questo permette di muovere il carattere sullo sfondo senza cancellarlo.

Programma 6.1A

```
org 42000 23760

DISABILITA LA TASTIERA

23760 F3          di
23761 18 18       jr START

UDG DATA

DATA1
CHR$144
defb 0 0 15 24 49 226 225 224

CHR$145
defb 24 60 255 255 153 255 231
defb 60

CHR$146
defb 0 0 240 24 140 71 135 7


START

PREDISPOSIZIONE UDG


23787 ED 5B 7B 5C ld de,(23675)
23791 21 D3 5C    ld hl,DATA1
23794 01 18 00    ld bc,24
23797 ED B0       ldir
23799 18 0A       jr BEGIN
```

```
PRINT CHR$. DATA AT x,y
INK 8 PAPER 8
CHR$144+CHR$145+CHR$146

DATA2
defb 22 0 0 16 8 17 8
defb 144 145 146
BEGIN


BORDER 5

23811 FD 36 0E 28 ld (iy+14),40

PAPER 1 INK 7

23815 FD 36 53 0F ld (iy+83),15

CLS

23819 3E 02       ld a,2
23821 CD 01 16    call 5633
23824 CD 6B 0D    call 3435

VISUALIZZA UNA STRINGA

23827 3E 02       ld a,2
23829 CD 01 16    call 5633
23832 11 F9 5C    ld de,DATA2
23835 01 0A 00    ld bc,10

23838 CD 3C 20    call 8252


L7

TRASFERISCE DATA2 ALLA MEMORIA

23841 11 00 5B    ld de,23296
23844 21 F9 5C    ld hl,DATA2
23847 01 0A 00    ld bc,10
23850 ED B0       ldir

LEGGE x,y DALLA MEMORIA

23852 2A 01 5B    ld hl,(23297)

CONTROLLA SE SIA STATO PREMUTO
UN TASTO
```

```
23855 01 FE EF     ld bc,61438
23858 ED 78        in a,(c)

23860 CB 47        bit 0,a
23862 20 01        jr nz,L1
23864 24           inc h
L1
23865 CB 4F        bit 1,a
23867 20 01        jr nz,L2
23869 25           dec h
L2
23870 01 FE F7     ld bc,63486
23873 ED 78        in a,(c)
23875 CB 47        bit 0,a
23877 20 01        jr nz,L3
23879 2D           dec l
L3
23880 01 FE FB     ld bc,64510
23883 ED 78        in a,(c)
23885 CB 47        bit 0,a
23887 20 01        jr nz,L4
23889 2C           inc l

x,y APPARTENGONO ALLO SCHERMO ?


L4
23890 7D           ld a,l
23891 FE 16        cp 22
23893 20 03        jr nz,L5
23895 2D           dec l
23896 18 05        jr L6
L5
23898 FE FF        cp 255
23900 20 01        jr nz,L6
23902 2C           inc l
L6
23903 7C           ld a,h
23904 FE 1E        cp 30
23906 20 03        jr nz,L8
23908 25           dec h
23909 18 05        jr L9
L8
23911 FE FF        cp 255
23913 20 01        jr nz,L9
23915 24           inc h

L9

NESSUNO SPOSTAMENTO ?
```

```
23916 ED 5B 01 5B ld de,(23297)
23920 AF          xor a
23921 E5          push hl
23922 ED 52       sbc hl,de
23924 E1          pop hl

SE NESSUNO SPOSTAMENTO, SALTA

23925 28 2C       jr z,L12

SALVA I NUOVI PARAMETRI x,y

23927 22 01 5B    ld (23297),hl

PONE OVER 1

23930 FD 36 57 03 ld (iy+87),3


CANCELLA LA VECCHIA POSIZIONE

23934 11 F9 5C    ld de,DATA2
23937 01 0A 00    ld bc,10
23940 CD 3C 20    call 8252

RISCRIVE NELLA NUOVA POSIZIONE

23943 11 00 5B    ld de,23296
23946 01 0A 00    ld bc,10
23949 CD 3C 20    call 8252

AGGIORNA DATA2

23952 11 F9 5C    ld de,DATA2
23955 21 00 5B    ld hl,23296
23958 01 0A 00    ld bc,10
23961 ED B0       ldir

LOOP DI RITARDO


23963 21 88 13    ld hl,5000
L11
23966 2B          dec hl
23967 7C          ld a,h
23968 B5          or l
23969 20 FB       jr nz,L11

CONTROLLA SE E' STATO PREMUTO
'SPACE'
```

```
L12
23971 01 FE 7F     ld bc,32766
23974 ED 78        in a,(c)
23976 CB 47        bit 0,a
23978 C2 21 5D     jp nz,L7

RIPRISTINA OVER 0

23981 FD 36 57 00 ld (iy+87),0


RIABILITA LA TASTIERA

23985 FB           ei
23986 C9           ret
```

Il programma 6.1A dimostra il principio sopra descritto. Un carattere simile ad una nave spaziale viene spostato sullo schermo usando i tasti 9/0 per sinistra/destra e 1/Q per su/giù. SPACE restituisce il controllo al BASIC. La nave spaziale può muoversi in due direzioni contemporaneamente e può volare su tutto lo schermo senza alterare alcun tipo di sfondo predisposto. Il carattere viene visualizzato in INK 8, PAPER 8, assumendo così i colori definiti dagli attributi di ogni singola zona dello schermo. La velocità di movimento è controllata dal loop di ritardo e viene eseguito un controllo per vedere se la nave spaziale è ferma. In tal caso non viene eseguita la routine di rivisualizzazione che farebbe lampeggiare la nave spaziale continuamente cancellata e rivisualizzata nello stesso punto. Potete modificare il programma per permettere alla nave spaziale di volare al di fuori dello schermo e riapparire sull'altro lato, semplicemente resettando y se la nave oltrepassa i limiti dello schermo.
Il movimento di più caratteri è analogo a quello appena descritto con un set di PRINT AT DATA a disposizione di ogni carattere. La routine in codice macchina è molto veloce, come si può vedere usando LD HL, 1 per il loop di ritardo: il televisore non sarà in grado di visualizzare tutte le posizioni assunte dal carattere nel suo movimento. Noterete che gran parte dei programmi necessita di un qualche loop di ritardo per fermare la visualizzazione per un istante prima di continuare.

Movimento dello sfondo a passi di un carattere

Mentre il movimento degli oggetti visualizzati richiede di conservare in memoria la posizione del carattere e tenere traccia degli spostamenti effettuati, il movimento dello sfondo richiede generalmente lo slittamento

dell'intero schermo (o di sue parti) in una certa direzione, senza bisogno di tener nota della effettiva posizione di ogni parte.
Questo movimento è normalmente in una delle quattro direzioni principali (su, giù, sinistra, destra) e generalmente avviene tanto sui caratteri che sui loro attributi. Slittando uno sfondo di un carattere abbiamo un'altra possibilità di scelta per quanto riguarda i caratteri usciti dallo schermo. Possiamo sia perderli e stampare spazi nella riga/colonna ottenuta all'altro estremo dello schermo, sia salvarli insieme ai loro attributi e rivisualizzarli ai posti corrispondenti sull'altro lato dello schermo, ottenendo quindi un effetto di avvolgimento che è estremamente utile nei programmi di giochi.
I seguenti programmi 6.1B–6.16 dimostrano il roll e lo scroll nelle quattro direzioni, prima degli attributi e poi dei caratteri visualizzati. Questi programmi sono facilmente modificabili per ottenere il roll o lo scroll di parti dello schermo, lasciando il resto intatto. Da notare la particolare configurazione dello schermo, che è diviso in tre blocchi di otto righe. Ogni riga viene successivamente divisa in otto linee di 32 byte ognuna. La memoria video inizia in riga 1 linea 1 e mappa i 32 byte di quella linea, quindi passa alla riga 2 linea 1, riga 3 linea 1 e così via fino a riga 8 linea 1, mappando 256 byte. Saltiamo indietro ora a riga 1 linea 2, riga 2 linea 2, riga 3 linea 2, eccetera, per finire con riga 8 linea 8, mappando così 2K byte.
Possiamo ora passare alle otto righe di mezzo che sono trattate in modo analogo e finalmente alle ultime otto righe, ancora configurate allo stesso modo. Questa configurazione del sistema rende un poco complicato il roll e lo scroll dei caratteri e richiede il movimento di 6K byte.

Programma 6.1B

```
org 42000 23760
SCROLL IN SU DEGLI ATTRIBUTI

23760 11 00 58     ld de,22528
23763 21 20 58     ld hl,22560
23766 01 E0 02     ld bc,736
23769 ED B0        ldir
23771 06 20        ld b,32
23773 3A 8D 5C     ld a,(23693)
L2
23776 12           ld (de),a
23777 13           inc de
23778 10 FC        djnz L2
23780 C9           ret
```

Programma 6.2

```
org 42000 23760
SCROLL IN GIU' DEGLI ATTRIBUTI

23760 11 FF 5A     ld de,23295
23763 21 DF 5A     ld hl,23263
23766 01 E0 02     ld bc,736
23769 ED B8        lddr
23771 06 20        ld b,32
23773 3A 8D 5C     ld a,(23693)
L7
23776 12           ld (de),a
23777 13           inc de
23778 10 FC        djnz L7
23780 C9           ret
```

Programma 6.3

```
org 42000 23760
SCROLL A SINISTRA DEGLI ATTRIBUTI

23760 06 18        ld b,24
23762 21 00 58     ld hl,22528
L12
23765 E5           push hl
23766 D1           pop de
23767 C5           push bc
23768 23           inc hl
23769 01 1F 00     ld bc,31
23772 ED B0        ldir
23774 3A 8D 5C     ld a,(23693)
23777 12           ld (de),a
23778 C1           pop bc
23779 10 F0        djnz L12
23781 C9           ret
```

Programma 6.4

```
org 42000 23760
SCROLL A DESTRA DEGLI ATTRIBUTI

23760 06 18        ld b,24
23762 21 FF 5A     ld hl,23295
L14
23765 E5           push hl
23766 D1           pop de
```

```
23767 C5          push bc
23768 2B          dec hl
23769 01 1F 00    ld bc,31
23772 ED B8       lddr
23774 3A 8D 5C    ld a,(23693)
23777 12          ld (de),a
23778 C1          pop bc
23779 10 F0       djnz L14
23781 C9          ret
```

Programma 6.5

```
org 42000 23760
ROLL IN SU DEGLI ATTRIBUTI

23760 21 00 58    ld hl,22528
23763 E5          push hl
23764 D1          pop de
23765 06 20       ld b,32
L16
23767 7E          ld a,(hl)
23768 F5          push af
23769 23          inc hl
23770 10 FB       djnz L16
23772 01 E0 02    ld bc,736
23775 ED B0       ldir
23777 06 20       ld b,32
L17
23779 F1          pop af
23780 2B          dec hl
23781 77          ld (hl),a
23782 10 FB       djnz L17

23784 C9          ret
```

Programma 6.6

```
org 42000 23760
ROLL IN GIU' DEGLI ATTRIBUTI

23760 21 FF 5A    ld hl,23295
23763 E5          push hl
23764 D1          pop de
23765 06 20       ld b,32
L24
23767 7E          ld a,(hl)
```

```
23768 F5          push af
23769 2B          dec hl
23770 10 FB       djnz L24
23772 01 E0 02    ld bc,736
23775 ED B8       lddr
23777 06 20       ld b,32
L25
23779 F1          pop af
23780 23          inc hl
23781 77          ld (hl),a
23782 10 FB       djnz L25

23784 C9          ret
```

Programma 6.7

```
org 42000 23760
ROLL A SINISTRA DEGLI ATTRIBUTI

23760 06 18       ld b,24
23762 21 00 58    ld hl,22528
L32
23765 E5          push hl
23766 D1          pop de
23767 C5          push bc
23768 7E          ld a,(hl)
23769 23          inc hl
23770 01 1F 00    ld bc,31
23773 ED B0       ldir
23775 12          ld (de),a
23776 C1          pop bc
23777 10 F2       djnz L32
23779 C9          ret
```

Programma 6.8

```
org 42000 23760
ROLL A DESTRA DEGLI ATTRIBUTI

23760 06 18       ld b,24
23762 21 FF 5A    ld hl,23295
L33
23765 E5          push hl
23766 D1          pop de
23767 C5          push bc
23768 7E          ld a,(hl)
```

```
23769 2B          dec hl
23770 01 1F 00    ld bc,31
23773 ED B8       lddr
23775 12          ld (de),a
23776 C1          pop bc
23777 10 F2       djnz L33
23779 C9          ret
```

Programma 6.9

```
org 42000 23760
ROLL A SINISTRA DELLO SCHERMO

23760 06 C0       ld b,192
23762 21 00 40    ld hl,16384
L1
23765 E5          push hl
23766 D1          pop de
23767 C5          push bc
23768 7E          ld a,(hl)
23769 23          inc hl
23770 01 1F 00    ld bc,31
23773 ED B0       ldir
23775 12          ld (de),a
23776 C1          pop bc
23777 10 F2       djnz L1
23779 C9          ret
```

Programma 6.10

```
org 42000 23760
ROLL A DESTRA DELLO SCHERMO

23760 06 C0       ld b,192
23762 21 FF 57    ld hl,22527
L1
23765 E5          push hl
23766 D1          pop de
23767 C5          push bc
23768 7E          ld a,(hl)
23769 2B          dec hl
23770 01 1F 00    ld bc,31
23773 ED B8       lddr
23775 12          ld (de),a
23776 C1          pop bc
23777 10 F2       djnz L1
23779 C9          ret
```

Programma 6.11

```
org 42000 23760
SCROLL A SINISTRA DELLO SCHERMO

23760 21 00 40     ld hl,16384
23763 AF           xor a
23764 06 C0        ld b,192
L13
23766 C5           push bc
23767 E5           push hl
23768 D1           pop de
23769 23           inc hl
23770 01 1F 00     ld bc,31
23773 ED B0        ldir
23775 12           ld (de),a
23776 C1           pop bc
23777 10 F3        djnz L13
23779 C9           ret
```

Programma 6.11A

```
org 42000 23760
L2
23760 21 00 40     ld hl,16384
23763 11 20 00     ld de,32
23766 06 C0        ld b,192
L1
23768 36 00        ld (hl),0
23770 19           add hl,de
23771 10 FB        djnz L1
23773 21 01 40     ld hl,16385
23776 11 00 40     ld de,16384
23779 01 FF 17     ld bc,6143
23782 ED B0        ldir
23784 EB           ex de,hl
23785 36 00        ld (hl),0
23787 C9           ret
```

Programma 6.12

```
org 42000 23760
SCROLL A DESTRA DELLO SCHERMO

23760 21 FF 57     ld hl,22527
23763 AF           xor a
```

```
23764 06 C0       ld b,192
L14
23766 C5          push bc
23767 E5          push hl
23768 D1          pop de
23769 2B          dec hl
23770 01 1F 00    ld bc,31
23773 ED B8       lddr
23775 12          ld (de),a
23776 C1          pop bc
23777 10 F3       djnz L14
23779 C9          ret
```

Programma 6.13

```
org 42000 23760
SCROLL IN SU DELLO SCHERMO

23760 11 00 40    ld de,16384
23763 21 20 40    ld hl,16416
23766 01 E0 07    ld bc,2016
23769 ED B0       ldir
23771 06 08       ld b,8
23773 21 00 48    ld hl,18432
23776 11 E0 40    ld de,16608
L3
23779 C5          push bc
23780 E5          push hl
23781 D5          push de
23782 01 20 00    ld bc,32
23785 ED B0       ldir
23787 D1          pop de
23788 E1          pop hl
23789 C1          pop bc
23790 14          inc d

23791 24          inc h
23792 10 F1       djnz L3
23794 11 00 48    ld de,18432
23797 21 20 48    ld hl,18464
23800 01 E0 07    ld bc,2016
23803 ED B0       ldir
23805 06 08       ld b,8
23807 21 00 50    ld hl,20480
23810 11 E0 48    ld de,18656
L4
23813 C5          push bc
23814 E5          push hl
```

```
23815 D5          push de
23816 01 20 00    ld bc,32
23819 ED B0       ldir
23821 D1          pop de
23822 E1          pop hl
23823 C1          pop bc
23824 14          inc d
23825 24          inc h
23826 10 F1       djnz L4

23828 11 00 50    ld de,20480
23831 21 20 50    ld hl,20512
23834 01 E0 07    ld bc,2016
23837 ED B0       ldir
23839 06 08       ld b,8
23841 21 E0 50    ld hl,20704
L5
23844 C5          push bc
23845 E5          push hl
23846 06 20       ld b,32
L6
23848 36 00       ld (hl),0
23850 23          inc hl
23851 10 FB       djnz L6
23853 E1          pop hl
23854 C1          pop bc
23855 24          inc h
23856 10 F2       djnz L5
23858 C9          ret
```

Programma 6.14

```
org 42000 23760
ROLL IN SU DELLO SCHERMO

23760 21 00 40    ld hl,16384
L18
23763 06 20       ld b,32
L19
23765 7E          ld a,(hl)
23766 F5          push af
23767 23          inc hl
23768 10 FB       djnz L19
23770 AF          xor a
23771 6F          ld l,a
23772 24          inc h
23773 7C          ld a,h
23774 FE 48       cp 72
```

```
23776 20 F1       jr nz,L18
23778 11 00 40    ld de,16384
23781 21 20 40    ld hl,16416
23784 01 E0 07    ld bc,2016

23787 ED B0       ldir
23789 06 08       ld b,8
23791 21 00 48    ld hl,18432
23794 11 E0 40    ld de,16608
L20
23797 C5          push bc
23798 E5          push hl
23799 D5          push de
23800 01 20 00    ld bc,32
23803 ED B0       ldir
23805 D1          pop de
23806 E1          pop hl
23807 C1          pop bc
23808 14          inc d
23809 24          inc h
23810 10 F1       djnz L20
23812 11 00 48    ld de,18432
23815 21 20 48    ld hl,18464
23818 01 E0 07    ld bc,2016
23821 ED B0       ldir
23823 06 08       ld b,8

23825 21 00 50    ld hl,20480
23828 11 E0 48    ld de,18656
L21
23831 C5          push bc
23832 E5          push hl
23833 D5          push de
23834 01 20 00    ld bc,32
23837 ED B0       ldir
23839 D1          pop de
23840 E1          pop hl
23841 C1          pop bc
23842 14          inc d
23843 24          inc h
23844 10 F1       djnz L21
23846 11 00 50    ld de,20480
23849 21 20 50    ld hl,20512
23852 01 E0 07    ld bc,2016
23855 ED B0       ldir
23857 21 FF 57    ld hl,22527
L22
23860 06 20       ld b,32
```

```
L23
23862 F1          pop af
23863 77          ld (hl),a
23864 2B          dec hl
23865 10 FB       djnz L23
23867 3E FF       ld a,255
23869 6F          ld l,a
23870 25          dec h
23871 7C          ld a,h
23872 FE 4F       cp 79
23874 20 F0       jr nz,L22
23876 C9          ret
23877 AF          xor a
23878 6F          ld l,a
23879 24          inc h
23880 7C          ld a,h
23881 FE 48       cp 72
23883 20 E7       jr nz,L22
23885 C9          ret
```

Programma 6.15

```
org 42000 23760
SCROLL IN GIU' DELLO SCHERMO

23760 11 FF 57    ld de,22527
23763 21 DF 57    ld hl,22495
23766 01 E0 07    ld bc,2016
23769 ED B8       lddr
23771 06 08       ld b,8
23773 21 E0 48    ld hl,18656
23776 11 00 50    ld de,20480
L8
23779 C5          push bc
23780 E5          push hl
23781 D5          push de
23782 01 20 00    ld bc,32
23785 ED B0       ldir
23787 D1          pop de
23788 E1          pop hl
23789 C1          pop bc
23790 14          inc d

23791 24          inc h
23792 10 F1       djnz L8
23794 11 FF 4F    ld de,20479
23797 21 DF 4F    ld hl,20447
23800 01 E0 07    ld bc,2016
```

```
23803 ED B8        lddr
23805 06 08        ld b,8
23807 21 E0 40     ld hl,16608
23810 11 00 48     ld de,18432
L9
23813 C5           push bc
23814 E5           push hl
23815 D5           push de
23816 01 20 00     ld bc,32
23819 ED B0        ldir
23821 D1           pop de
23822 E1           pop hl
23823 C1           pop bc
23824 14           inc d
23825 24           inc h
23826 10 F1        djnz L9

23828 11 FF 47     ld de,18431
23831 21 DF 47     ld hl,18399
23834 01 E0 07     ld bc,2016
23837 ED B8        lddr
23839 21 00 40     ld hl,16384
L10
23842 06 20        ld b,32
L11
23844 36 00        ld (hl),0
23846 23           inc hl
23847 10 FB        djnz L11
23849 AF           xor a
23850 6F           ld l,a
23851 24           inc h
23852 7C           ld a,h
23853 FE 48        cp 72
23855 20 F1        jr nz,L10
23857 C9           ret
```

Programma 6.16

```
org 42000 23760
ROLL IN GIU' DELLO SCHERMO

23760 21 FF 57     ld hl,22527
L26
23763 06 20        ld b,32
L27
23765 7E           ld a,(hl)
23766 F5           push af
23767 2B           dec hl
```

```
23768 10 FB       djnz L27
23770 3E FF       ld a,255
23772 6F          ld l,a
23773 25          dec h
23774 7C          ld a,h
23775 FE 4F       cp 79
23777 20 F0       jr nz,L26
23779 11 FF 57    ld de,22527
23782 21 DF 57    ld hl,22495
23785 01 E0 07    ld bc,2016

23788 ED B8       lddr
23790 06 08       ld b,8
23792 21 E0 48    ld hl,18656
23795 11 00 50    ld de,20480
L28
23798 C5          push bc
23799 E5          push hl
23800 D5          push de
23801 01 20 00    ld bc,32
23804 ED B0       ldir
23806 D1          pop de
23807 E1          pop hl
23808 C1          pop bc
23809 14          inc d
23810 24          inc h
23811 10 F1       djnz L28
23813 11 FF 4F    ld de,20479
23816 21 DF 4F    ld hl,20447
23819 01 E0 07    ld bc,2016
23822 ED B8       lddr
23824 06 08       ld b,8

23826 21 E0 40    ld hl,16608
23829 11 00 48    ld de,18432
L29
23832 C5          push bc
23833 E5          push hl
23834 D5          push de
23835 01 20 00    ld bc,32
23838 ED B0       ldir
23840 D1          pop de
23841 E1          pop hl
23842 C1          pop bc
23843 14          inc d
23844 24          inc h
23845 10 F1       djnz L29
23847 11 FF 47    ld de,18431
23850 21 DF 47    ld hl,18399
23853 01 E0 07    ld bc,2016
```

```
23856 ED B8        lddr
23858 21 00 40     ld hl,16384
L30
23861 06 20        ld b,32

L31
23863 F1           pop af
23864 77           ld (hl),a
23865 23           inc hl
23866 10 FB        djnz L31
23868 AF           xor a
23869 6F           ld l,a
23870 24           inc h
23871 7C           ld a,h
23872 FE 48        cp 72
23874 20 F1        jr nz,L30
23876 C9           ret
```

Programma 6.17

```
500 DATA 8,79,5,2,"FERMA IL NASTRO"
510 DATA 20,167,6,3,"MOVIMENTO"
520 DATA 0,130,2,4,"________"
525 DATA 15,15,1,2,"PREMI UN TASTO"
```

```
 550 DATA 41,170,1,2,"ISTRUZIONI"
 600 BORDER 4: PAPER 6: INK 0: CLS
 605 RESTORE 500
 610 FOR a=1 TO 4
 620 INK a: GO SUB 8000
 630 PAUSE 50: NEXT a
 640 PAUSE 0: CLS
 642 INK 2: GO SUB 8000
 645 PRINT  INK 0;AT 3,0;"Questo programma  d
imostra come si possa muovere lo sfondo e gli
oggetti su esso visualizzati."
 646 PRINT '"La tabella mostra i tasti usati
e le relative funzioni."
 647 PRINT
 650 PRINT "          SFONDO  "
 655 PRINT "       ROLL  ¦ SCROLL     OGGETTO "
 660 PRINT "SU      1    ¦     Q          7"
 670 PRINT "GIU'    2    ¦     W          8"
 680 PRINT "SIN.    3    ¦     E          9"
 690 PRINT "DESTRA 4    ¦     R          0"
 695 PRINT '"SPACE restituisce il controllo
al BASIC"
 700 RESTORE 525: INK 2: GO SUB 8000
 705 IF INKEY$<>"" THEN  GO TO 705
 710 IF INKEY$="" THEN  GO TO 710
 715 POKE 30046,0: POKE 30047,0
 720 RANDOMIZE USR 30001
 725 CLS : RESTORE 730: PAPER 8: INK 0: GO SU
B 8000
 730 DATA 0,80,3,1,"DIGITA'GOTO 715'per ricom
inciare"
 740 STOP
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 30000: LOAD "udg"CODE USR "a": LOA
D ""CODE : GO TO 500
```

Programma 6.18

```
org 40000 30001
30001 F3          di
30002 CD D0 5C    call 23760
30005 18 18       jr START
DATA1
defb 0 0 15 24 49 226 225 224
defb 24 60 255 255 153 255 231
defb 60 0 0 240 24 140 71 135 7
START
```

```
30031 ED 5B 7B 5C ld de,(23675)
30035 21 37 75    ld hl,DATA1
30038 01 18 00    ld bc,24
30041 ED B0       ldir
30043 18 0A       jr BEGIN
DATA2
defb 22 0 0 16 8 17 8 144 145
defb 146
BEGIN
30055 11 5D 75    ld de,DATA2
30058 01 0A 00    ld bc,10

30061 CD 3C 20    call 8252
30064 11 00 5B    ld de,23296
30067 21 5D 75    ld hl,DATA2
30070 01 0A 00    ld bc,10
30073 ED B0       ldir
MOVE
30075 2A 01 5B    ld hl,(23297)
30078 01 FE EF    ld bc,61438
30081 ED 78       in a,(c)
30083 FE FF       cp 255
30085 28 5D       jr z,LP12
30087 CB 47       bit 0,a
30089 20 01       jr nz,LP1
30091 24          inc h
LP1
30092 CB 4F       bit 1,a
30094 20 01       jr nz,LP2
30096 25          dec h
LP2
30097 CB 57       bit 2,a
30099 20 01       jr nz,LP3

30101 2C          inc l
LP3
30102 CB 5F       bit 3,a
30104 20 01       jr nz,LP4
30106 2D          dec l
LP4
30107 7D          ld a,l
30108 FE 16       cp 22
30110 20 03       jr nz,LP5
30112 2D          dec l
30113 18 05       jr LP6
LP5
30115 FE FF       cp 255
30117 20 01       jr nz,LP6
30119 2C          inc l
LP6
```

```
30120 7C          ld a,h
30121 FE 1E       cp 30
30123 20 03       jr nz,LP7
30125 25          dec h
30126 18 05       jr LP8

LP7
30128 FE FF       cp 255
30130 20 01       jr nz,LP8
30132 24          inc h
LP8
30133 ED 5B 01 5B ld de,(23297)
30137 AF          xor a
30138 E5          push hl
30139 ED 52       sbc hl,de
30141 E1          pop hl
30142 28 24       jr z,LP12
30144 22 01 5B    ld (23297),hl
OVER1
30147 FD 36 57 03 ld (iy+87),3
30151 11 5D 75    ld de,DATA2
30154 01 0A 00    ld bc,10
30157 CD 3C 20    call 8252
30160 11 00 5B    ld de,23296
30163 01 0A 00    ld bc,10
30166 CD 3C 20    call 8252
30169 11 5D 75    ld de,DATA2

30172 21 00 5B    ld hl,23296
30175 01 0A 00    ld bc,10
30178 ED B0       ldir
LP12
30180 01 FE F7    ld bc,63486
30183 ED 78       in a,(c)
30185 E6 0F       and 15
30187 28 3E       jr z,SCR
30189 FE 03       cp 3
30191 28 3A       jr z,SCR
30193 FE 0C       cp 12
30195 28 36       jr z,SCR
30197 FE 0F       cp 15
30199 28 32       jr z,SCR
30201 F5          push af
30202 FD 36 57 03 ld (iy+87),3
30206 11 5D 75    ld de,DATA2
30209 01 0A 00    ld bc,10
30212 CD 3C 20    call 8252
30215 F1          pop af
30216 CB 47       bit 0,a
```

```
30218 F5          push af
30219 CC C9 77    call z,UROLL
30222 F1          pop af
30223 CB 4F       bit 1,a
30225 F5          push af
30226 CC 56 78    call z,DROLL
30229 F1          pop af
30230 CB 57       bit 2,a
30232 F5          push af
30233 CC E3 78    call z,LROLL
30236 F1          pop af
30237 CB 5F       bit 3,a
30239 CC F7 78    call z,RROLL
30242 11 5D 75    ld de,DATA2
30245 01 0A 00    ld bc,10
30248 CD 3C 20    call 8252
SCR
30251 01 FE FB    ld bc,64510
30254 ED 78       in a,(c)
30256 E6 0F       and 15
30258 28 3E       jr z,DELAY

30260 FE 03       cp 3
30262 28 3A       jr z,DELAY
30264 FE 0C       cp 12
30266 28 36       jr z,DELAY
30268 FE 0F       cp 15
30270 28 32       jr z,DELAY
30272 F5          push af
30273 FD 36 57 03 ld (iy+87),3
30277 11 5D 75    ld de,DATA2
30280 01 0A 00    ld bc,10
30283 CD 3C 20    call 8252
30286 F1          pop af
30287 CB 47       bit 0,a
30289 F5          push af
30290 CC 8A 76    call z,USCR
30293 F1          pop af
30294 CB 4F       bit 1,a
30296 F5          push af
30297 CC 01 77    call z,DSCR
30300 F1          pop af
30301 CB 57       bit 2,a

30303 F5          push af
30304 CC 77 77    call z,LSCR
30307 F1          pop af
30308 CB 5F       bit 3,a
30310 CC A0 77    call z,RSCR
30313 11 5D 75    ld de,DATA2
```

```
30316 01 0A 00     ld bc,10
30319 CD 3C 20     call 8252
DELAY
30322 21 10 27     ld hl,10000
L1
30325 2B           dec hl
30326 7C           ld a,h
30327 B5           or l
30328 20 FB        jr nz,L1
30330 01 FE 7F     ld bc,32766
30333 ED 78        in a,(c)
30335 CB 47        bit 0,a
30337 C2 7B 75     jp nz,MOVE
30340 FD 36 57 00  ld (iy+87),0
30344 FB           ei

30345 C9           ret
USCR
30346 11 00 58     ld de,22528
30349 21 20 58     ld hl,22560
30352 01 E0 02     ld bc,736
30355 ED B0        ldir
30357 06 20        ld b,32
30359 3A 8D 5C     ld a,(23693)
L2
30362 12           ld (de),a
30363 13           inc de
30364 10 FC        djnz L2
30366 11 00 40     ld de,16384
30369 21 20 40     ld hl,16416
30372 01 E0 07     ld bc,2016
30375 ED B0        ldir
30377 06 08        ld b,8
30379 21 00 48     ld hl,18432
30382 11 E0 40     ld de,16608
L3
30385 C5           push bc

30386 E5           push hl
30387 D5           push de
30388 01 20 00     ld bc,32
30391 ED B0        ldir
30393 D1           pop de
30394 E1           pop hl
30395 C1           pop bc
30396 14           inc d
30397 24           inc h
30398 10 F1        djnz L3
30400 11 00 48     ld de,18432
30403 21 20 48     ld hl,18464
```

```
30406 01 E0 07    ld bc,2016
30409 ED B0       ldir
30411 06 08       ld b,8
30413 21 00 50    ld hl,20480
30416 11 E0 48    ld de,18656
L4
30419 C5          push bc
30420 E5          push hl
30421 D5          push de

30422 01 20 00    ld bc,32
30425 ED B0       ldir
30427 D1          pop de
30428 E1          pop hl
30429 C1          pop bc
30430 14          inc d
30431 24          inc h
30432 10 F1       djnz L4
30434 11 00 50    ld de,20480
30437 21 20 50    ld hl,20512
30440 01 E0 07    ld bc,2016
30443 ED B0       ldir
30445 06 08       ld b,8
30447 21 E0 50    ld hl,20704
L5
30450 C5          push bc
30451 E5          push hl
30452 06 20       ld b,32
L6
30454 36 00       ld (hl),0
30456 23          inc hl

30457 10 FB       djnz L6
30459 E1          pop hl
30460 C1          pop bc
30461 24          inc h
30462 10 F2       djnz L5
30464 C9          ret
DSCR
30465 11 FF 5A    ld de,23295
30468 21 DF 5A    ld hl,23263
30471 01 E0 02    ld bc,736
30474 ED B8       lddr
30476 06 20       ld b,32
30478 3A 8D 5C    ld a,(23693)
L7
30481 12          ld (de),a
30482 13          inc de
30483 10 FC       djnz L7
30485 11 FF 57    ld de,22527
```

```
30488 21 DF 57     ld hl,22495
30491 01 E0 07     ld bc,2016
30494 ED B8        lddr

30496 06 08        ld b,8
30498 21 E0 48     ld hl,18656
30501 11 00 50     ld de,20480
L8
30504 C5           push bc
30505 E5           push hl
30506 D5           push de
30507 01 20 00     ld bc,32
30510 ED B0        ldir
30512 D1           pop de
30513 E1           pop hl
30514 C1           pop bc
30515 14           inc d
30516 24           inc h
30517 10 F1        djnz L8
30519 11 FF 4F     ld de,20479
30522 21 DF 4F     ld hl,20447
30525 01 E0 07     ld bc,2016
30528 ED B8        lddr
30530 06 08        ld b,8
30532 21 E0 40     ld hl,16608

30535 11 00 48     ld de,18432
L9
30538 C5           push bc
30539 E5           push hl
30540 D5           push de
30541 01 20 00     ld bc,32
30544 ED B0        ldir
30546 D1           pop de
30547 E1           pop hl
30548 C1           pop bc
30549 14           inc d
30550 24           inc h
30551 10 F1        djnz L9
30553 11 FF 47     ld de,18431
30556 21 DF 47     ld hl,18399
30559 01 E0 07     ld bc,2016
30562 ED B8        lddr
30564 21 00 40     ld hl,16384
L10
30567 06 20        ld b,32
L11

30569 36 00        ld (hl),0
30571 23           inc hl
```

```
30572 10 FB       djnz L11
30574 AF          xor a
30575 6F          ld l,a
30576 24          inc h
30577 7C          ld a,h
30578 FE 48       cp 72
30580 20 F1       jr nz,L10
30582 C9          ret
LSCR
30583 06 18       ld b,24
30585 21 00 58    ld hl,22528
L12
30588 E5          push hl
30589 D1          pop de
30590 C5          push bc
30591 23          inc hl
30592 01 1F 00    ld bc,31
30595 ED B0       ldir
30597 3A 8D 5C    ld a,(23693)

30600 12          ld (de),a
30601 C1          pop bc
30602 10 F0       djnz L12
30604 21 00 40    ld hl,16384
30607 AF          xor a
30608 06 C0       ld b,192
L13
30610 C5          push bc
30611 E5          push hl
30612 D1          pop de
30613 23          inc hl
30614 01 1F 00    ld bc,31
30617 ED B0       ldir
30619 12          ld (de),a
30620 C1          pop bc
30621 10 F3       djnz L13
30623 C9          ret
RSCR
30624 06 18       ld b,24
30626 21 FF 5A    ld hl,23295
L14

30629 E5          push hl
30630 D1          pop de
30631 C5          push bc
30632 2B          dec hl
30633 01 1F 00    ld bc,31
30636 ED B8       lddr
30638 3A 8D 5C    ld a,(23693)
30641 12          ld (de),a
```

```
30642 C1          pop bc
30643 10 F0       djnz L14
30645 21 FF 57    ld hl,22527
30648 AF          xor a
30649 06 C0       ld b,192
L14
30651 C5          push bc
30652 E5          push hl
30653 D1          pop de
30654 2B          dec hl
30655 01 1F 00    ld bc,31
30658 ED B8       lddr
30660 12          ld (de),a

30661 C1          pop bc
30662 10 DD       djnz L14
30664 C9          ret
UROLL
30665 21 00 58    ld hl,22528
30668 E5          push hl
30669 D1          pop de
30670 06 20       ld b,32
L16
30672 7E          ld a,(hl)
30673 F5          push af
30674 23          inc hl
30675 10 FB       djnz L16
30677 01 E0 02    ld bc,736
30680 ED B0       ldir
30682 06 20       ld b,32
L17
30684 F1          pop af
30685 2B          dec hl
30686 77          ld (hl),a
30687 10 FB       djnz L17

30689 21 00 40    ld hl,16384
L18
30692 06 20       ld b,32
L19
30694 7E          ld a,(hl)
30695 F5          push af
30696 23          inc hl
30697 10 FB       djnz L19
30699 AF          xor a
30700 6F          ld l,a
30701 24          inc h
30702 7C          ld a,h
30703 FE 48       cp 72
30705 20 F1       jr nz,L18
```

```
30707 11 00 40     ld de,16384
30710 21 20 40     ld hl,16416
30713 01 E0 07     ld bc,2016
30716 ED B0        ldir
30718 06 08        ld b,8
30720 21 00 48     ld hl,18432
30723 11 E0 40     ld de,16608

L20
30726 C5           push bc
30727 E5           push hl
30728 D5           push de
30729 01 20 00     ld bc,32
30732 ED B0        ldir
30734 D1           pop de
30735 E1           pop hl
30736 C1           pop bc
30737 14           inc d
30738 24           inc h
30739 10 F1        djnz L20
30741 11 00 48     ld de,18432
30744 21 20 48     ld hl,18464
30747 01 E0 07     ld bc,2016
30750 ED B0        ldir
30752 06 08        ld b,8
30754 21 00 50     ld hl,20480
30757 11 E0 48     ld de,18656
L21
30760 C5           push bc

30761 E5           push hl
30762 D5           push de
30763 01 20 00     ld bc,32
30766 ED B0        ldir
30768 D1           pop de
30769 E1           pop hl
30770 C1           pop bc
30771 14           inc d
30772 24           inc h
30773 10 F1        djnz L21
30775 11 00 50     ld de,20480
30778 21 20 50     ld hl,20512
30781 01 E0 07     ld bc,2016
30784 ED B0        ldir
30786 21 FF 57     ld hl,22527
L22
30789 06 20        ld b,32
L23
30791 F1           pop af
30792 77           ld (hl),a
30793 2B           dec hl
```

```
30794 10 FB        djnz L23
30796 3E FF        ld a,255
30798 6F           ld l,a
30799 25           dec h
30800 7C           ld a,h
30801 FE 4F        cp 79
30803 20 F0        jr nz,L22
30805 C9           ret
DROLL
30806 21 FF 5A     ld hl,23295
30809 E5           push hl
30810 D1           pop de
30811 06 20        ld b,32
L24
30813 7E           ld a,(hl)
30814 F5           push af
30815 2B           dec hl
30816 10 FB        djnz L24
30818 01 E0 02     ld bc,736
30821 ED B8        lddr
30823 06 20        ld b,32

L25
30825 F1           pop af
30826 23           inc hl
30827 77           ld (hl),a
30828 10 FB        djnz L25
30830 21 FF 57     ld hl,22527
L26
30833 06 20        ld b,32
L27
30835 7E           ld a,(hl)
30836 F5           push af
30837 2B           dec hl
30838 10 FB        djnz L27
30840 3E FF        ld a,255
30842 6F           ld l,a
30843 25           dec h
30844 7C           ld a,h
30845 FE 4F        cp 79
30847 20 F0        jr nz,L26
30849 11 FF 57     ld de,22527
30852 21 DF 57     ld hl,22495

30855 01 E0 07     ld bc,2016
30858 ED B8        lddr
30860 06 08        ld b,8
30862 21 E0 48     ld hl,18656
30865 11 00 50     ld de,20480
L28
```

```
30868 C5          push bc
30869 E5          push hl
30870 D5          push de
30871 01 20 00    ld bc,32
30874 ED B0       ldir
30876 D1          pop de
30877 E1          pop hl
30878 C1          pop bc
30879 14          inc d
30880 24          inc h
30881 10 F1       djnz L28
30883 11 FF 4F    ld de,20479
30886 21 DF 4F    ld hl,20447
30889 01 E0 07    ld bc,2016
30892 ED B8       lddr

30894 06 08       ld b,8
30896 21 E0 40    ld hl,16608
30899 11 00 48    ld de,18432
L29
30902 C5          push bc
30903 E5          push hl
30904 D5          push de
30905 01 20 00    ld bc,32
30908 ED B0       ldir
30910 D1          pop de
30911 E1          pop hl
30912 C1          pop bc
30913 14          inc d
30914 24          inc h
30915 10 F1       djnz L29
30917 11 FF 47    ld de,18431
30920 21 DF 47    ld hl,18399
30923 01 E0 07    ld bc,2016
30926 ED B8       lddr
30928 21 00 40    ld hl,16384
L30

30931 06 20       ld b,32
L31
30933 F1          pop af
30934 77          ld (hl),a
30935 23          inc hl
30936 10 FB       djnz L31
30938 AF          xor a
30939 6F          ld l,a
30940 24          inc h
30941 7C          ld a,h
30942 FE 48       cp 72
30944 20 F1       jr nz,L30
```

```
30946 C9          ret
LROLL
30947 06 D8       ld b,216
30949 21 00 40    ld hl,16384
L32
30952 E5          push hl
30953 D1          pop de
30954 C5          push bc
30955 7E          ld a,(hl)

30956 23          inc hl
30957 01 1F 00    ld bc,31
30960 ED B0       ldir
30962 12          ld (de),a
30963 C1          pop bc
30964 10 F2       djnz L32
30966 C9          ret
RROLL
30967 06 D8       ld b,216
30969 21 FF 5A    ld hl,23295
L33
30972 E5          push hl
30973 D1          pop de
30974 C5          push bc
30975 7E          ld a,(hl)
30976 2B          dec hl
30977 01 1F 00    ld bc,31
30980 ED B8       lddr
30982 12          ld (de),a
30983 C1          pop bc
30984 10 F2       djnz L33

30986 C9          ret
```

Da notare l'uso estensivo fatto in queste routine delle istruzioni di trasferimento blocchi LDIR e LDDR e che, dopo l'esecuzione di queste istruzioni, i registri DE e HL contengono gli indirizzi finali +1 (LDIR) o −1 (LDDR). Questa caratteristica viene sfruttata per evitare di dover ricaricare HL o DE con altri indirizzi per le successive routine del programma. Il programma di scroll in su degli attributi (Programma 6.1B), per esempio, sfrutta il fatto che dopo l'esecuzione dell'istruzione LDIR, la coppia di registri DE contiene l'indirizzo dell'attributo INIZIO DELLA LINEA 24 ed è quindi inutile reinizializzare DE con lo stesso indirizzo. Gli attributi presenti sulla linea 24 vengono rimpiazzati con gli attributi permanenti di schermo contenuti nella variabile di sistema ATTR P (23693).
Per dimostrare ogni tipo di movimento ho scritto il programma 6.17. Poiché il programma è lungo circa 1000 byte, non è pratico immagazzinarlo

tutto in una REM; l'ho congegnato quindi per usarlo al di sopra della RAMTOP, con indirizzo iniziale 30001 (programma 6.18). Il codice macchina nella REM è estratto dal programma dei numeri casuali (programma 4.1) ed è usato per predisporre lo sfondo. Il programma è autoesplicante e chiama le varie routine in funzione dei tasti premuti. Gli attributi vengono spostati contemporaneamente ai caratteri. Potete, se volete, muovere lo sfondo e l'oggetto diagonalmente mentre non viene generato alcun movimento se vengono premuti contemporaneamente due tasti di comando per direzioni opposte. È possibile ritornare al BASIC premendo il tasto SPACE. Da notare che il movimento verticale non è eccessivamente elegante e non del tutto soddisfacente a causa delle transizioni da una zona di schermo all'altra. Il programma, in effetti, immagazzina momentaneamente la linea 1 nella linea 8 prima di effettuare lo scroll della linea 9 in linea 8. I programmi di scroll e di roll in su e in giù (programmi 6.13–6.16) sono stati scritti in una forma molto comprensibile, senza subroutine tali da renderli dipendenti dalla posizione. Per ottenere un movimento più dolce, queste routine potranno (e dovranno) essere trattate in modi diversi, cioè effettuando lo scroll di una riga per volta, piuttosto che lo scroll di una linea di ogni riga per volta, così come sono. Per mostrare come ottenere questo, ho riscritto il programma di scroll in su dello schermo (programma 6.19A). Come potete vedere, esso sfrutta una subroutine ed una sub-subroutine ed è alquanto più complesso. Potreste riscrivere la routine di scroll in giù dello schermo in maniera analoga, come esercizio per una completa comprensione del funzionamento.

Programma 6.19A

```
org 42000 23760
SCROLL IN SU DELLO SCHERMO
VERSIONE MODIFICATA

23760 21 00 40     ld hl,16384
23763 CD 01 5D     call SECT
23766 EB           ex de,hl
23767 21 00 48     ld hl,18432
23770 E5           push hl
23771 CD 15 5D     call UP
23774 E1           pop hl
23775 CD 01 5D     call SECT
23778 EB           ex de,hl
23779 21 00 50     ld hl,20480
23782 E5           push hl
23783 CD 15 5D     call UP
23786 E1           pop hl
23787 CD 01 5D     call SECT
```

```
23790 21 E0 50     ld hl,20704
L4

23793 E5           push hl
23794 06 20        ld b,32
L3
23796 36 00        ld (hl),0
23798 23           inc hl
23799 10 FB        djnz L3
23801 E1           pop hl
23802 24           inc h
23803 7C           ld a,h
23804 FE 58        cp 88
23806 20 F1        jr nz,L4
23808 C9           ret
SECT
23809 06 07        ld b,7
L1
23811 C5           push bc
23812 01 20 00     ld bc,32
23815 E5           push hl
23816 D1           pop de
23817 09           add hl,bc
23818 E5           push hl

23819 D5           push de
23820 CD 15 5D     call UP
23823 D1           pop de
23824 E1           pop hl
23825 C1           pop bc
23826 10 EF        djnz L1
23828 C9           ret
UP
23829 06 08        ld b,8
L2                    .
23831 C5           push bc
23832 E5           push hl
23833 D5           push de
23834 01 20 00     ld bc,32
23837 ED B0        ldir
23839 D1           pop de
23840 E1           pop hl
23841 C1           pop bc
23842 14           inc d
23843 24           inc h
23844 10 F1        djnz L2

23846 C9           ret
```

Se confrontate i programmi 6.13 e 6.19A noterete che abbiamo ridotto l'occupazione di memoria da 99 a 87 byte; ma potremmo ulteriormente compattarli. Per mostrare come diminuire occupazione di memoria e tempo di esecuzione, ottenendo in più un'altra interessante possibilità, esaminiamo il programma di roll in su (programma 6.14), che occupa 117 byte. Attualmente, il programma gestisce ogni riga di caratteri linea per linea ed ha il difetto di trasferire temporaneamente riga 1 in riga 8 e riga 9 in riga 17. Potremmo riscrivere questo programma sulla falsariga del programma 6.21 evitando così il problema; prendiamo in considerazione però un approccio completamente differente in grado di fornirci un programma lungo solo 76 byte.
Invece di gestire lo schermo una riga per volta, possiamo scrivere un programma che esegua il roll di una colonna per volta. La routine sarebbe poi ripetuta per 32 colonne. Questo vi permetterebbe di scegliere quali colonne desiderate trattare, tanto in blocco che singolarmente e, ancora più importante, non richiede 256 byte di memoria per immagazzinare la riga in cima allo schermo (come il programma 6.14), ma solo 8 byte per il singolo carattere in cima alla colonna. Il programma 6.19 B mostra il procedimento.

Programma 6.19 B

```
org 42000 23760
ROLL IN SU DI 'n' COLONNE

ESEMPIO

ROLL DELLE COLONNE 5 - 14

23760 AF            xor a

LD DE,16383+COLONNA INIZIALE

PRIMA COLONNA = 1

23761 11 04 40      ld de,16388

LD B,Numero di colonne

23764 06 0A         ld b,10
L5
23766 D5            push de

23767 C5            push bc

SALVA SULLO STACK I CARATTERI
DELLA LINEA SUPERIORE
```

```
23768 D5          push de
23769 E1          pop hl
23770 06 08       ld b,8
L6
23772 7E          ld a,(hl)
23773 F5          push af
23774 24          inc h
23775 10 FB       djnz L6

TRASFERISCE 7 LINEE

L4
23777 06 08       ld b,8
L2
23779 C5          push bc
23780 21 20 00    ld hl,32

23783 19          add hl,de
23784 E5          push hl

TRASFERISCE 8 BYTE

23785 06 08       ld b,8
L1
23787 7E          ld a,(hl)
23788 12          ld (de),a
23789 24          inc h
23790 14          inc d
23791 10 FA       djnz L1
23793 D1          pop de
23794 C1          pop bc
23795 10 EE       djnz L2

TRASFERISCE LA ZONA DI SCHERMO

23797 21 20 07    ld hl,1824
23800 19          add hl,de
23801 7C          ld a,h


CONTROLLA IL RAGGIUNGIMENTO
DEL FUORI SCHERMO, cioe'
dell'inizio di ATTRIBUTES

23802 FE 58       cp 88
23804 28 0C       jr z,END
23806 E5          push hl
23807 06 08       ld b,8
L3
23809 7E          ld a,(hl)
```

```
23810 12          ld (de),a
23811 24          inc h
23812 14          inc d
23813 10 FA       djnz L3
23815 D1          pop de
23816 18 D7       jr L4

RIPRENDE DALLO STACK I CARAT-
TERI E LI PONE NELLA RIGA
INFERIORE DELLO SCHERMO


END
23818 11 20 00    ld de,32
23821 ED 52       sbc hl,de
23823 06 08       ld b,8
L7
23825 F1          pop af
23826 77          ld (hl),a
23827 25          dec h
23828 10 FB       djnz L7

CAMBIA COLONNA E RIPETE

23830 C1          pop bc
23831 D1          pop de
23832 13          inc de
23833 10 BB       djnz L5
23835 C9          ret
```

Come potete notare, il programma 6.19 B inizia con l'indirizzo del byte in cima alla colonna iniziale posto nella coppia di registri DE e il numero di colonne su cui eseguire il roll contenuto nel registro B. Così com'è, il programma è completamente autocontenuto ed è posizionabile in qualunque zona della memoria. Ma possiamo ancora risparmiare memoria. Credereste che è possibile ridurre il programma 6.19 B a solo 58 byte e aggiungere la possibilità di definire le righe su cui eseguire il roll, ottenendo quindi una routine in grado di operare su finestre? Il programma 6.19 C mostra come fare. Il risparmio di memoria è ottenuto eseguendo il controllo della fine di una zona di schermo tramite l'istruzione BIT 0,H. Se BIT 0,H è non 0, allora siamo arrivati alla fine di una zona e invece di aggiungere 32 a DE per incrementare di una linea l'interno della zona, noi aggiungiamo 1824 per passare alla prossima linea nella zona successiva. Grazie a questo metodo, possiamo definire su quante righe effettuare il roll dal numero di volte per cui è ripetuto il loop L4. La RIGA/COLONNA iniziale è definita dall'indirizzo contenuto nella coppia di registri DE (23671/3).

Per calcolare questo indirizzo usate la seguente formula:

16384+(1824*INT(RIGA/8)) = ((RIGA−INT(RIGA/8))*32)+COLONNA

In questa formula vanno usati i numeri di RIGA/COLONNA dello Spectrum, cioè 1ª riga = 0, 1ª colonna = 0.

Esempio: RIGA 17 COLONNA 5
= 16384+(1824*2)+((17−6)*32)+5
= 16384+3648+32+5
= 20069

Programma 6.19C

```
org 42000 23760
ROLL IN SU DI UNA FINESTRA

ESEMPIO
COLONNE 4 - 10
RIGHE 5 - 18

23760 AF            xor a

LD DE,INDIRIZZO INIZ. SCHERMO
cioe'
INDIRIZZO INIZIALE COLONNA/RIGA

23761 11 83 40      ld de,16515

LD B,Numero di colonne

23764 06 07         ld b,7
L6
23766 D5            push de

23767 C5            push bc
23768 D5            push de
23769 E1            pop hl
23770 06 08         ld b,8
L1
23772 7E            ld a,(hl)
23773 F5            push af
23774 24            inc h
23775 10 FB         djnz L1

LD B,Numero di righe-1!

23777 06 0D         ld b,13
L4
```

```
23779 C5          push bc
23780 21 20 00    ld hl,32
23783 19          add hl,de

CONTROLLA IL CAMBIO
DELLA ZONA DI SCHERMO


23784 CB 44       bit 0,h
23786 28 04       jr z,L2
23788 21 20 07    ld hl,1824
23791 19          add hl,de
L2
23792 E5          push hl
23793 06 08       ld b,8
L3
23795 7E          ld a,(hl)
23796 12          ld (de),a
23797 14          inc d
23798 24          inc h
23799 10 FA       djnz L3
23801 D1          pop de
23802 C1          pop bc
23803 10 E6       djnz L4
23805 06 08       ld b,8
L5
23807 25          dec h
23808 F1          pop af
23809 77          ld (hl),a

23810 10 FB       djnz L5
23812 C1          pop bc
23813 D1          pop de
23814 D1          pop de
23815 13          inc de
23816 10 CC       djnz L6
23818 C9          ret
```

Per eseguire il roll dell'intero schermo, DE dovrà contenere 16384, il registro B (prima di L6) 32 e, sempre il registro B (prima di L4), 23. Questo programma è facilmente adattabile per ottenere routine di scroll in su o in giù e di roll in giù.

Movimento di pixel

Questo tipo di movimento di oggetti su uno sfondo non è del tutto indicato per programmi di giochi, poiché tende ad essere piuttosto lento. La ve-

locità è ridotta di un fattore 8: se, ad esempio, dovete eseguire un controllo di collisione di un carattere su un ostacolo, sarà necessario effettuare la verifica su ogni bit della forma del carattere usando l'equivalente in codice macchina di POINT x, y, per vedere, prima della rivisualizzazione, se la nuova posizione è libera. Così, se un carattere ha dimensione 16×16 pixel, supponendo anche di controllare solo i pixel perimetrali, saranno necessarie forse 100 chiamate alla routine POINT prima di spostare il carattere. Tutto questo richiede tempo, il che vuol dire riduzione drastica della velocità di movimento. È tuttavia possibile, comunque, il movimento dello sfondo, un pixel alla volta, purché gli attributi (che non possono essere spostati di un pixel) rimangano fissi.

Nel programma dimostrativo PIXELDEMO, disponibile su nastro, si dimostra la lentezza del movimento di pixel, nonché la presenza di un certo sfarfallio a causa del movimento dello sfondo. È necessario rimuovere il pallone dallo schermo durante il movimento dello sfondo ed eseguire un controllo di POINT prima di rivisualizzarlo. Il tempo impiegato è notevole, pur essendo stato mantenuto il più breve possibile, grazie alla scelta del pallone che, necessitando solo del contorno, riduce al minimo il numero dei controlli di POINT e il numero di chiamate a PLOT.

Il principio adottato è lo stesso sfruttato nel movimento a passi di un carattere, con le posizioni di PLOT x, y immagazzinate come dati, la presenza di un set duplicato di dati aggiornato e controllato, il pallone cancellato usando OVER 1 e rivisualizzato grazie ai dati duplicati. Il pallone è in grado di muoversi oltre gli estremi sinistro e destro dello schermo come in un avvolgimento continuo. Questo è in effetti necessario per evitare le frecce sulla riga inferiore.

Il programma è in bianco e nero perché, se fossero stati usati più di due colori, il pallone avrebbe cambiato colore nel movimento da una zona di attributi ad un'altra col rischio di diventare indistinguibile, e anche perché così appare ugualmente bene su un televisore in bianco e nero.

Il programma dimostra i principi che stanno alla base del roll dello schermo a sinistra e a destra: ogni byte di carattere viene ruotato a sinistra (RLA) o a destra (RRA) e se un bit viene trasferito nel CARRY, esso entra a far parte del byte di carattere adiacente. Maggiori chiarimenti saranno forniti nel Capitolo 9.

Per dimostrare il roll in giù di un pixel, il seguente programma (programma 6.20) traccia un'onda sinusoidale e, disegnata una linea in cima allo schermo, esegue un roll in giù finché non appare una strada scura in lento movimento. Questo sistema può essere usato in un programma tipo corsa automobilistica. Esso serve inoltre a dimostrare come attribuire valori non interi al passo di un loop (vedi Capitolo 7).

Programma 6.20

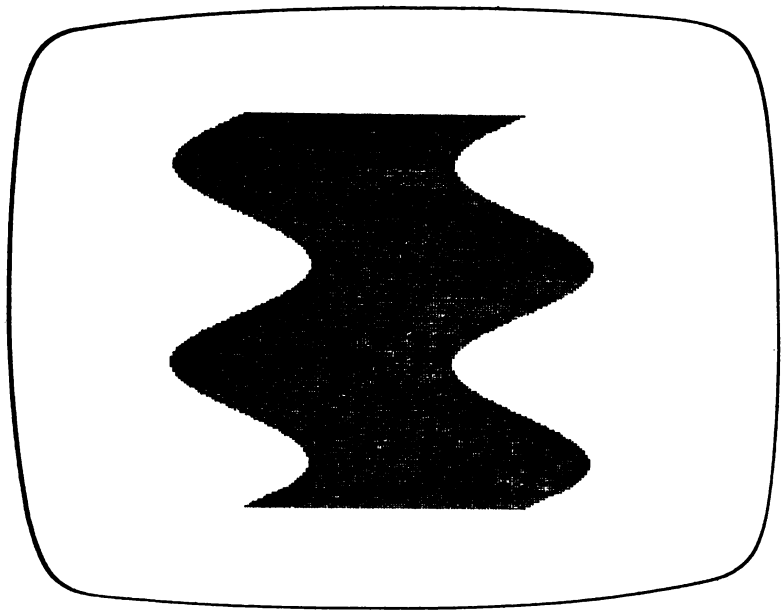
```
org 42000 23760

10 FOR a=0 TO 4*PI STEP PI/44
20 PLOT 100+SIN a*25,175
30 DRAW 100,0
40 REM Call roll in giu' di
       un pixel
50 NEXT a

AZZERA a ( su calculator )

23760 EF          rst 40
defb 160 56

SALVA il valore di a in DATA

L1
23763 11 35 5D    ld de,DATA
23766 01 05 00    ld bc,5
23769 ED B0       ldir


STACK 4*PI ( su calc )
23771 EF          rst 40
defb 163 56
23774 34          inc (hl)
23775 34          inc (hl)
23776 34          inc (hl)

SUB a-4*PI

23777 EF          rst 40
defb 3 56

CONTROLLA IL PRIMO BYTE DI
MANTISSA PER LA RISPOSTA -ve

23780 23          inc hl
23781 7E          ld a,(hl)
23782 CB 7F       bit 7,a
23784 20 04       jr nz,L2


CLEAR CALCULATOR se -ve

23786 CD F1 2B    call 11249

RITORNA AL BASIC
```

```
23789 C9          ret

CLEAR CALCULATOR

L2
23790 CD F1 2B    call 11249
23793 CD 3A 5D    call ROLL

STACK DATA a

23796 CD 24 5D    call STACK

100+SIN a*25=x

23799 EF          rst 40

sine
defb 31
stack 25
defb 52 53 72
mult
defb 4
stack 100
defb 52 55 72
add
defb 15
stack 175 (=y)
defb 52 56 47 56

CALL PLOT x,y

23813 CD DC 22    call 8924

SALVA H'L'

23816 D9          exx
23817 E5          push hl

23818 D9          exx

LD B,0  LD C,100

23819 01 64 00    ld bc,100

LD DE, +ve +ve

23822 11 01 01    ld de,257

CALL DRAW
```

```
23825 CD BA 24     call 9402

RIPRENDE H'L'

23828 D9           exx
23829 E1           pop hl
23830 D9           exx

STACK DATA a


23831 CD 24 5D     call STACK

a+PI/44
23834 EF           rst 40
stack PI/2
defb 163
stack 22
defb 52 53 48
divide
defb 5
add
defb 15
fine routine calc
defb 56
23842 18 AF        jr L1

STACK a SUB

STACK
23844 21 35 5D     ld hl,DATA

23847 ED 5B 65 5C  ld de,(23653)
23851 01 05 00     ld bc,5
23854 ED B0        ldir
23856 ED 53 65 5C  ld (23653),de
23860 C9           ret

IMMAGAZZINAMENTO DATA a

DATA
defb 0 0 0 0 0

ROLL IN GIU'

ROLL
23866 06 20        ld b,32
23868 21 A0 57     ld hl,22432
L3
23871 7E           ld a,(hl)
```

```
23872 F5          push af
23873 23          inc hl
23874 10 FB       djnz L3

23876 21 BF 56    ld hl,22207
23879 11 BF 57    ld de,22463
23882 06 06       ld b,6
23884 CD 86 5D    call DOWN
23887 21 E0 4F    ld hl,20448
23890 11 00 50    ld de,20480
23893 01 20 00    ld bc,32
23896 ED B0       ldir
23898 21 FF 4E    ld hl,20223
23901 11 FF 4F    ld de,20479
23904 06 08       ld b,8
23906 CD 86 5D    call DOWN
23909 21 E0 47    ld hl,18400
23912 11 00 48    ld de,18432
23915 01 20 00    ld bc,32
23918 ED B0       ldir
23920 21 FF 46    ld hl,18175
23923 11 FF 47    ld de,18431
23926 06 08       ld b,8
23928 CD 86 5D    call DOWN
23931 06 20       ld b,32

23933 21 1F 40    ld hl,16415
L4
23936 F1          pop af
23937 77          ld (hl),a
23938 2B          dec hl
23939 10 FB       djnz L4
23941 C9          ret
DOWN
23942 C5          push bc
23943 06 07       ld b,7
L5
23945 C5          push bc
23946 E5          push hl
23947 E5          push hl
23948 01 20 00    ld bc,32
23951 ED B8       lddr
23953 D1          pop de
23954 E1          pop hl
23955 25          dec h
23956 C1          pop bc
23957 10 F2       djnz L5

23959 7C          ld a,h
23960 C6 08       add a,8
```

```
23962 67          ld h,a
23963 7D          ld a,l
23964 D6 20       sub 32
23966 6F          ld l,a
23967 E5          push hl
23968 E5          push hl
23969 01 20 00    ld bc,32
23972 ED B8       lddr
23974 E1          pop hl
23975 D1          pop de
23976 25          dec h
23977 C1          pop bc
23978 10 DA       djnz DOWN
23980 C9          ret
```



La routine di roll in giù di un pixel, alla fine del programma 6.20, è stata riscritta per facilitarne la comprensione. Può essere ulteriormente modificata sulla falsariga del programma 6.19 C per ottenere il roll in giù di un pixel di una finestra, rendendola nel contempo indipendente dalla posizione di caricamento e usando meno memoria. Così com'è, essa occupa 115 byte (più 32 per immagazzinare la riga inferiore) ma, riscritta come programma 6.20A, richiede solo 44 byte (+1 per il byte inferiore della colonna).

Programma 6.20A

```
org 42000 23760
ROLL IN GIU' DI UNA FINESTRA

ESEMPIO

COLONNE 4 - 10
LINEE 4 - 39

REM LINEA 1/COLONNA 1 = ANGOLO
SUPERIORE SINISTRO

23760 11 83 46     ld de,18051

LD B,Colonne

23763 06 07        ld b,7
L2
23765 C5           push bc
23766 D5           push de
23767 1A           ld a,(de)

23768 F5           push af
23769 D5           push de
23770 E1           pop hl
23771 25           dec h

LD B,Linee-1 !

23772 06 23        ld b,35
L1
23774 E5           push hl
23775 7E           ld a,(hl)
23776 12           ld (de),a
23777 D1           pop de
23778 7C           ld a,h
23779 25           dec h
23780 E6 07        and 7
23782 20 0A        jr nz,L3
23784 7D           ld a,l
23785 D6 20        sub 32
23787 6F           ld l,a
23788 38 04        jr c,L3

23790 7C           ld a,h
23791 C6 08        add a,8
23793 67           ld h,a
L3
23794 10 EA        djnz L1
```

```
23796 F1          pop af
23797 12          ld (de),a
23798 D1          pop de
23799 C1          pop bc
23800 13          inc de
23801 10 DA       djnz L2
23803 C9          ret
```

Il registro B contiene (prima di L2) il numero di colonne su cui eseguire il roll e (prima di L1) il numero di linee meno 1. Il registro DE contiene l'indirizzo di schermo del byte inferiore della prima colonna, che è calcolato usando:

LET A = LINEA (0 è la prima linea)
LET B = INT (A/64)
LET C = A−(b*64)
LET D = INT (C/8)
LET E = C−(D*8)
LET F = COLONNA (0 è la prima colonna)
INDIRIZZO = 16384+(1824*B)+(256*E)+(32*(D−1))+F
Esempio: LINEA 38 : COLONNA 3
A = 38 : B = 0 : C = 38 : D = 3 : E = 6 : F = 3
INDIRIZZO = 16384+(1824*0)+(256*6)+(32*4)+3 = 18051

Poiché questa formula è piuttosto lunga, potreste scrivere un programma in codice macchina in grado di eseguire questo calcolo. Iniziate con il numero LINEA/COLONNA contenuto nella coppia di registri HL e finite con la coppia di registri DE contenente l'indirizzo di schermo, in modo che possiate direttamente cedere il controllo alla routine ROLL (o in alternativa sfruttate una tabella che mostri i singoli indirizzi di LINEA/COLONNA nella memoria video).
Perché non provare a scrivere una routine simile? Potrebbe essere utile in molti programmi, ad esempio per effettuare il POKE di caratteri direttamente sullo schermo. Il programma 6.20 B presenta un metodo ed occupa solo 29 byte (che sarebbero solo 28 se LINEA/COLONNA fosse caricato in HL usando LD HL, 034 Fh). Ho inserito questo programma in coda al programma 6.20A.
Studiate questo programma sino alla sua completa comprensione poiché esso è molto utile per capire il file di visualizzazione.

Programma 6.20B

```
org 42000 23760
ROLL IN GIU' DI UNA FINESTRA

ESEMPIO

COLONNE 3 - 9
LINEE 56 - 79

REM LINEA 0/COLONNA 0 = ANGOLO
SUPERIORE SINISTRO

LD H,Linea

23760 26 4F        ld h,79

LD L,Colonna

23762 2E 03        ld l,3
23764 11 00 40     ld de,16384
23767 7C           ld a,h

23768 E6 07        and 7
23770 B2           or d
23771 57           ld d,a
23772 7C           ld a,h
23773 E6 C0        and 192
23775 1F           rra
23776 1F           rra
23777 1F           rra
23778 82           add a,d
23779 57           ld d,a
23780 7C           ld a,h
23781 E6 38        and 56
23783 17           rla
23784 17           rla
23785 5F           ld e,a
23786 7D           ld a,l
23787 83           add a,e
23788 5F           ld e,a

LD B,Colonne


23789 06 07        ld b,7
L2
23791 C5           push bc
23792 D5           push de
23793 1A           ld a,(de)
```

```
23794 F5          push af
23795 D5          push de
23796 E1          pop hl
23797 25          dec h

LD B,Linee-1 !

23798 06 17       ld b,23
L1
23800 E5          push hl
23801 7E          ld a,(hl)
23802 12          ld (de),a
23803 D1          pop de
23804 7C          ld a,h
23805 25          dec h
23806 E6 07       and 7

23808 20 0A       jr nz,L3
23810 7D          ld a,l
23811 D6 20       sub 32
23813 6F          ld l,a
23814 38 04       jr c,L3
23816 7C          ld a,h
23817 C6 08       add a,8
23819 67          ld h,a
L3
23820 10 EA       djnz L1
23822 F1          pop af
23823 12          ld (de),a
23824 D1          pop de
23825 C1          pop bc
23826 13          inc de
23827 10 DA       djnz L2
23829 C9          ret
```

Questi due programmi per movimento di pixel (6.20A e 6.20B) mostrano come eseguire il roll a sinistra, a destra e in giù dello sfondo. Lascio a voi il compito di scrivere programmi per ottenere il roll in su e lo scroll. Ricordatevi che non esiste un solo metodo corretto in codice macchina. Ognuno avrà un suo metodo personale di esecuzione, scelta dei registri, uso delle istruzioni PUSH/POP o per il normale immagazzinamento dei valori; tutto è accettabile e soddisfacente se il programma funziona. Molti dei programmi fin qui presentati possono essere riscritti in modo da risparmiare memoria o tempo di esecuzione ma potrebbero non essere altrettanto chiari per i principianti.
Esiste un'altra utile routine della ROM che non è accessibile dal BASIC; essa effettua lo scroll di n linee con i relativi attributi, con n>2 e <25. Il programma 6.21 mostra come usare la routine per svuotare le linee da

16 a 24 incluse (9 linee), chiamando 9 volte la routine. Da notare che il numero di linee su cui eseguire l'operazione viene contato a partire dalla linea 24. Questa routine può essere usata al posto di quella inserita nel programma 6.17. Il registro B contiene il numero di linee da processare −1; si provvede quindi a chiamare la routine 3584. La linea inferiore viene sovrastampata con spazi, usando i colori contenuti in ATTR P.

Programma 6.21

```
org 42000 23760
SCROLL IN SU DI n LINEE

23760 06 09       ld b,9
L1
23762 C5          push bc
23763 06 08       ld b,8
23765 CD 00 0E    call 3584
23768 C1          pop bc
23769 10 F7       djnz L1
23771 C9          ret
```

Presentiamo infine una routine che esegue il roll dello schermo a sinistra o a destra a passi di mezzo carattere. Questo programma, benché corto, è piuttosto difficile e sfrutta l'istruzione di rotazione a sinistra (o a destra) decimale. Questa istruzione implica la gestione a *nibble* (mezzo byte = quattro bit) del registro A e del contenuto della cella indirizzata dalla coppia di registri HL. RLD trasferisce i bit 0−3, (HL) nei bit 7−4, (HL); i bit 7−4, (HL) nei bit 0−3, A e i bit 0−3, A nei bit 0−3, (HL). Essa non varia alcun flag.
Come potete vedere nel programma 6.22 questa istruzione è usata su ogni linea di schermo, sfruttando il registro A per immagazzinare il nibble destro e per trasferirlo nel nibble sinistro del carattere adiacente. Il nibble sinistro del primo carattere viene cancellato, poiché il registro A contiene, all'inizio di ogni linea, 0. Alla fine di ogni linea, il registro A contiene il nibble sinistro che sarà trasferito al di fuori dello schermo e aggiunto al byte iniziale della linea per dare un effetto di avvolgimento continuo. Se desiderate uno sfondo che cambi di continuo, potete immagazzinare un nibble destro di ricambio sotto forma di dati, da aggiungere al byte destro, invece del nibble spostato al di fuori dello schermo. Lascio a voi il compito della stesura di una routine di roll a destra (un suggerimento: partite all'inizio del file di visualizzazione).

Programma 6.22

```
org 42000 23760
ROLL DELLO SCHERMO A SINISTRA
DI MEZZO CARATTERE

23760 F3          di

Inizia dalla fine del video

23761 21 FF 57    ld hl,22527

Numero di linee dello schermo

23764 06 C4       ld b,196
L1
23766 C5          push bc

Reset a=0

23767 AF          xor a


Numero di byte/linea

23768 06 20       ld b,32

Salva l'inizio della linea

23770 E5          push hl

Ruota a sinistra a,(hl)
istruzione nibble

L2
23771 ED 6F       rld
23773 2B          dec hl
23774 10 FB       djnz L2

Pone in de l'inizio della linea

23776 D1          pop de

Salva l'attuale pos. di schermo


23777 EB          ex de,hl

Somma il nibble LH al byte RH
```

```
23778 86          add a,(hl)
23779 77          ld (hl),a

Riprende la pos. di schermo
in hl

23780 EB          ex de,hl
23781 C1          pop bc
23782 10 EE       djnz L1
23784 FB          ei
23785 C9          ret
```

Capitolo

Musica ed effetti sonori

7

In questo capitolo vedremo come migliorare la grafica e l'animazione con suoni e musica. È prima necessario esaminare il modo in cui lo Spectrum immagazzina i numeri in virgola mobile, nonché come convertire un numero in virgola mobile nella sua rappresentazione compressa. Se le vostre conoscenze matematiche non sono eccessivamente profonde, non preoccupatevi troppo della teoria e sfruttate l'aiuto che vi può dare lo Spectrum.
In parole povere ogni numero, decimale, intero, positivo o negativo, può essere rappresentato sotto forma di 5 byte, usando una particolare procedura. Il primo byte viene chiamato esponente, i rimanenti 4 mantissa.
La conversione di un numero nella sua rappresentazione in 5 byte richiede quattro passaggi:

1. Esprimere il numero nella sua forma binaria.

   Per esempio 11.375 diventerà 1011.011
   Ricordate che 0.1 binario = 1/2
   0.01 binario = 1/4
   ecc.

2. Trovare l'esponente.
   Spostate il punto decimale a sinistra o a destra finché esso non si trova alla sinistra del primo 1 nel numero binario e contate il numero di spostamenti effettuato. Nel nostro esempio l'esponente sarà pari a 128 più il numero di spostamenti.
   Se gli spostamenti sono verso sinistra, il numero da aggiungere sarà positivo. Se gli spostamenti sono verso destra, il numero da aggiungere sarà negativo.

Nel nostro esempio l'esponente è uguale a 128+4 = 132 e restiamo con .1011011

3. Alterare il bit a destra del punto decimale per indicare il segno del numero originale.
   Se il numero è positivo il bit viene posto a 0, se il numero è negativo il bit viene lasciato ad 1.
   Nel nostro esempio il numero era positivo ed avremo quindi: .0011011
4. Trovare la mantissa.
   Scartate il punto decimale e dividete le rimanenti cifre binarie in 4 byte, aggiungendo 0 dove necessario per ottenere 32 bit.
   00110110 00000000 00000000 00000000
   e riconvertite ogni byte al corrispondente numero decimale.
   Nel nostro esempio
   11.375 = 132 54 0 0 0

Se a questo punto avete un po' di confusione in testa, chiedete aiuto allo Spectrum. Lo Spectrum dispone di un utile metodo di immagazzinamento di semplici variabili: esse vengono memorizzate nella loro forma a 5 byte in virgola mobile, immediatamente dopo il nome della variabile. Tutto quello che ci serve è la scrittura di un semplice programma che introduca un numero e quindi legga i dati immagazzinati nella locazione di memoria per ricavare la rappresentazione a 5 byte.

Programma NUMERI

```
   5 GO TO 5000
  10 REM    PROGRAMMA PER VISUALIZZARE LA RAP
PRESENTAZIONE IN VIRGOLA MOBILE DI QUALUNQUE
NUMERO
  20 PRINT AT 3,0;"QUESTO BREVE PROGRAMMA FOR
NISCE LA  RAPPRESENTAZIONE IN VIRGOLA MOBILE
 DI  QUALUNQUE  NUMERO,  DECIMALE,  INTERO  O
  NEGATIVO. AI NUMERI INTERI VIENE SOMMATO  P
ER LA CONVERSIONE IL  DECIMALE 0.00000000001
CHE NON NE ALTERA IL VALORE.
    "
  25 PRINT '''"IL  NUMERO  VIENE  ANCHE FORNIT
ORAPPRESENTATO IN FORMA COMPRESSAUTILE PER IL
  LITERAL 34H (52d) "
  40 RETURN
  100  INPUT  "NUMERO:";TAB  9;n
  200  LET  n$="NUMERO:"
  300  LET  b$="VIRGOLA MOBILE   :"
  400  LET  c$="FORMA COMPRESSA :"
  500  LET  d$="□"

  1003 IF N<>0 THEN  GO TO 1010
  1005 PRINT n$;TAB 9;d$; PAPER 6;0'b$;TAB 9;d$
```

```
;"O  ";d$;"O  ";d$;"O  ";d$;"O  ";d$;"O"
1006 PRINT c$;TAB 9;d$;"O  ";d$;"176";d$;"O"
1007 GO TO 100
1010 LET N=N+.00000000001
1020 PRINT n$;TAB 9;d$; PAPER 6;N
1025 PRINT b$;
1030 LET X=PEEK 23627+256*PEEK 23628
1040 FOR A=1 TO 5
1050 PRINT TAB 5+4*a;d$;PEEK (X+A);
1060 NEXT A
1065 PRINT '
1070 DIM a(5)
1075 LET q=0
1076 PRINT c$;
1080 FOR b=5 TO 1 STEP -1
1085 LET a(b)=PEEK ((PEEK 23627+256*PEEK 2362
8)+b)
1086 IF b=2 THEN  GO TO 1100
1090 IF q=0 AND a(b)=0 THEN  NEXT b
1100 LET q=q+1
1110 NEXT b
1120 LET q=q-2
1125 LET x=(a(1)-80)
1126 IF x>=64 THEN  GO TO 2000
1130 LET e=x+(q*64)
1140 PRINT TAB 9;d$;e;
1150 FOR b=1 TO q+1
1160 PRINT TAB 9+b*4;d$;a(1+b);
1170 NEXT b
1180 PRINT
1190 GO TO 100
2000 PRINT TAB 9;d$;q*64;TAB 13;d$;a(1)-80;
2010 FOR b=1 TO q+1
2020 PRINT TAB 13+b*4;d$;a(1+b);
2030 NEXT b
2040 PRINT : GO TO 100
5000 DATA 8,79,5,2,"FERMA IL NASTRO"
5100 DATA 30,167,7,4,"NUMERI"
5200 DATA 15,130,2,4,"_______"
5250 DATA 15,15,1,2,"PREMI UN TASTO"
5500 DATA 41,170,1,2,"ISTRUZIONI"
6000 BORDER 4: PAPER 6: INK 0: CLS
6010 LET n=0
6050 RESTORE 5000
6100 FOR a=1 TO 4
6200 INK a: GO SUB 8000
6300 PAUSE 50: NEXT a
6400 PAUSE 0: CLS
6420 INK 2: GO SUB 8000
6450 INK 0: PAPER 7: GO SUB 10
```

```
7000 RESTORE 5250: INK 2: GO SUB 8000
7005 IF INKEY$<>"" THEN  GO TO 7005
7006 IF INKEY$="" THEN  GO TO 7006
7010 CLS : GO TO 100
8000 READ x,y,h,w,a$
8010 LET c=USR 32393
8020 RETURN
9998 CLEAR 32334: LOAD "udgs"CODE USR "a",168
: LOAD ""CODE : GO TO 5000
```

Il programma visualizza la rappresentazione in virgola mobile a 5 byte e la forma compressa di ogni numero. Maggiori dettagli sulla forma compressa saranno dati in seguito. Da notare che 0 viene trattato come un caso speciale in cui tutti i byte sono 0 e che 0.5 e 1/2 danno risultati differenti. Vi starete presumibilmente chiedendo cosa tutto questo abbia a che fare con la musica e i suoni. Il comando BASIC BEEP sfrutta lo stack di calcolo e ogni numero in esso memorizzato deve essere nella forma a 5 byte. Possiamo ora vedere come convertire la durata e il tono del comando BASIC BEEP in codice macchina. È necessario memorizzare i valori della durata e del tono nella loro forma a 5 byte in cima allo stack di calcolo e quindi chiamare la routine BEEP dall'indirizzo 1016 (03F8 in esadecimale).
Se prendiamo, ad esempio, BEEP.1,11, sfruttando il programma NUMERI possiamo convertire .1 e 11 nella loro forma a 5 byte:

.1 = 125 76 204 204 204
11 = 132 48 0 0 0

Dobbiamo ora trovare un sistema per immagazzinare questi 10 byte consecutivamente nello stack di calcolo. Un metodo consiste nel trovare l'indirizzo iniziale cui i dati vanno trasferiti per mezzo di un PEEK nella variabile di sistema STACKEND e quindi sfruttare una istruzione LDIR come nel programma 7.1.

Programma 7.1

```
org 42000 23760

BEEP .1,11


TROVA STACKEND

23760 ED 5B 65 5C ld de,(23653)
23764 21 E4 5C    ld hl,DATA
```

```
23767 01 0A 00     ld bc,10
23770 ED B0        ldir

AGGIORNA STACKEND

23772 ED 53 65 5C ld (23653),de

CALL BEEP

23776 CD F8 03     call 1016
23779 C9           ret

DATA
0.1 (rappresentazione a 5 byte)
defb 125 76 204 204 204

11 (rappresentazione a 5 byte)
defb 132 48 0 0 0
```

Il reset è necessario per porre in STACKEND il nuovo indirizzo della cima dello stack. Ricordate che all'uscita dall'istruzione LDIR, HL e DE contengono l'indirizzo dell'ultimo byte trasferito più 1. La routine BEEP chiamata dall'indirizzo 1016, preleva i due valori in cima allo stack e li elabora per fornire i valori richiesti di durata e tono. Lo stack viene quindi resettato per ottenere un sicuro ritorno al BASIC.
Un altro modo di porre sullo stack numeri interi consiste nell'usare LD A,n e CALL STACK A, all'indirizzo 11560 (2D28h) per i numeri compresi tra 0 e 255; per i numeri tra 0 e 65535 è possibile usare LD BC,nn e la subroutine STACK BC all'indirizzo 11563 (2D2Bh). È così possibile riscrivere il programma 7.1 nella seguente forma:

Programma 7.2

```
org 42000 23760

BEEP .1,11


TROVA STACKEND

23760 ED 5B 65 5C ld de,(23653)
23764 21 E9 5C    ld hl,DATA
23767 01 05 00    ld bc,5
23770 ED B0       ldir

AGGIORNA STACKEND
```

```
23772 ED 53 65 5C ld (23653),de
23776 3E 0B       ld a,11

STACK a

23778 CD 28 2D    call 11560


CALL BEEP

23781 CD F8 03    call 1016
23784 C9          ret
DATA
0.1 (rappresentazione a 5 byte)
defb 125 76 204 204 204
```

Infine, possiamo convertire i numeri in virgola mobile nella forma compressa ed usare il Literal di calcolo 52 (34h) per immagazzinare i dati nella forma compressa. I passi per convertire i numeri a 5 byte in forma compressa sono i seguenti:

1. Iniziando dal 5° byte, rimuovete tutti gli zeri fino a raggiungere un numero diverso da zero o il secondo byte.
2. Il quoziente è il numero di byte rimanenti meno 1.
3. Sottraete 80 (50h) dall'esponente e, se la differenza è maggiore o uguale a 64 (40h), passate all'istruzione 6.
4. Cambiate l'esponente a (esponente -80)+(quoziente$*64$).
5. Passate all'istruzione 7.
6. L'esponente diventa due byte. Il primo byte è il quoziente $*64$; il secondo byte è l'esponente -80.
7. Piazzate i byte rimanenti dopo l'esponente così modificato.

Per convertire un numero in forma compressa nella rappresentazione a 5 byte, usate la seguente procedura:

1. Dividete il primo byte per 64.
2. Se c'è un resto, l'esponente sarà pari a 80 più il resto.
3. Se non c'è resto l'esponente sarà 80 più il secondo byte.
4. Il quoziente viene usato per trovare quanti ulteriori byte seguono l'esponente. Il numero di questi byte è pari a 1 più il quoziente.

Ad esempio, 11 = 52 48 in forma compressa.
52 : 64 = 0 con il resto di 52
quoziente = 0
byte successivi = 1
Poiché c'era un resto, esponente = 52+80 = 132

1 byte ulteriore = 48
Completate i 5 byte con 0 0 0
Quindi 52 48 = 132 48 0 0 0

Se ancora vi sembra troppo complesso, sfruttate la conversione eseguita dal programma NUMERI. Sfruttando numeri in forma compressa è possibile convertire BEEP.1,11 in codice macchina:

Programma 7.3

```
org 42000 23760

BEEP .1,11

USA CALCULATOR

23760 EF            rst 40

STACK 0.1 (FORMA COMPRESSA)

defb 52 237 76 204 204 204

STACK 11 (FORMA COMPRESSA)

defb 52 52 48

FINE CALC

defb 56


CALL BEEP

23771 CD F8 03      call 1016
23774 C9            ret
```

Nel programma 7.3 RST 40 (28h) chiama la routine CALCULATOR e defb 56 (38h) è il Literal END CALC.
La fanfara del programma CROSS è una combinazione dei tre metodi: immagazzinare i numeri sotto forma di DATA, sfruttare lo stack A e sfruttare il Literal 52 (34h) per immagazzinare i numeri in forma compressa. Fortunatamente, in questo programma, i byte di mantissa della forma compressa dei decimali 0.1, 0.8 e 0.05 sono uguali, quindi sarà necessario immagazzinare come DATA solo il byte di esponente.

Programma 7.4

```
org 42000 23760

FANFARA

10 FOR a=1 TO 8
20 READ b,c
30 BEEP b,c
40 NEXT a
50 DATA .1,11,.1,11,.8,16,.05,
11,.05,16,.05,11,.05,16,1,20

23760 06 07        ld b,7
23762 21 00 5D     ld hl,DATA
LOOP
23765 7E           ld a,(hl)
23766 32 DE 5C     ld (VALUE),a
23769 23           inc hl
23770 C5           push bc
23771 E5           push hl
23772 EF           rst 40

defb 52
VALUE
defb 0 76 204 204 204 56
23780 E1           pop hl
23781 7E           ld a,(hl)
23782 E5           push hl
23783 CD 28 2D     call 11560
23786 CD F8 03     call 1016
23789 E1           pop hl
23790 C1           pop bc
23791 23           inc hl
23792 10 E3        djnz LOOP
23794 3E 01        ld a,1
23796 CD 28 2D     call 11560
23799 3E 14        ld a,20
23801 CD 28 2D     call 11560
23804 CD F8 03     call 1016
23807 C9           ret
DATA
defb 237 11 237 11 240 16 236
defb 11 236 16 236 11 236 16
```

Abbiamo finora visto come convertire qualunque comando BEEP in codice macchina. Questo va bene per la musica, ma non produrrà niente di simile ad un decente effetto sonoro. Per questo tipo di suoni avremo bisogno di sfruttare un diverso entry point della routine BEEP, 949

(03B5h). Una volta rimossi dallo stack, i valori di durata e tono vengono manipolati, finché il valore di tono sarà contenuto nel registro HL e la durata nel registro DE. È possibile scrivere un programma che dia un tono decrescente, incrementando il valore immagazzinato in HL e chiamando successivamente la routine BEEP in un loop FOR/NEXT.
Il programma 7.5 mostra come agire.

Programma 7.5

```
org 42000 23760

SUONO 1

23760 06 FF       ld b,255
23762 21 01 00    ld hl,1
LOOP
23765 11 01 00    ld de,1
23768 E5          push hl
23769 C5          push bc

CALL BEEP

23770 CD B5 03    call 949
23773 C1          pop bc
23774 E1          pop hl
23775 23          inc hl
23776 10 F3       djnz LOOP
23778 C9          ret
```

Noterete che, eseguendo il programma 7.5, via via che cambia gradualmente il tono, cambia anche la durata. Questo perché, in effetti, la durata è funzione dei valori contenuti tanto in HL che in DE. Per controllare anche la durata, alla diminuzione del valore contenuto in DE deve corrispondere un aumento del valore contenuto in HL. Siamo quindi arrivati al programma 7.6.

Programma 7.6

```
org 42000 23760

SUONO 2

23760 06 64       ld b,100
23762 21 01 00    ld hl,1
23765 11 65 00    ld de,101
LOOP
```

```
23768 D5          push de
23769 E5          push hl
23770 C5          push bc
23771 CD B5 03    call 949
23774 C1          pop bc
23775 E1          pop hl
23776 D1          pop de
23777 23          inc hl
23778 1B          dec de
23779 10 F3       djnz LOOP
23781 C9          ret
```

Vi presentiamo un ultimo metodo di produrre suoni con lo Spectrum; si tratta di mandare segnali direttamente all'altoparlante, commutandolo alto o basso in modo analogo a quanto fa la routine ROM. Questo metodo non è troppo pratico per note musicali, perché avremmo bisogno di conoscere la frequenza di commutazione per ogni nota e arriveremmo infine a duplicare la routine ROM. Si rivela invece utile in programmi di giochi per ottenere un rumore bianco per esplosioni, collisioni, ...
L'istruzione di controllo è OUT (254),A. Questa istruzione controlla non solo l'altoparlante ma anche il colore temporaneo della cornice nella maniera seguente:

1. Se bit 0,A = 1 allora blu è acceso
   Se bit 0,A = 0 allora blu è spento
2. Se bit 1,A = 1 allora rosso è acceso
   Se bit 1,A = 0 allora rosso è spento
3. Se bit 2,A = 1 allora verde è acceso
   Se bit 2,A = 0 allora verde è spento
4. Se bit 4,A = 1 allora l'altoparlante è "alto"
   Se bit 4,A = 0 allora l'altoparlante è "basso".

Come potete vedere, settando e resettando ad intervalli regolari il bit 4, si produrrà una nota in funzione degli intervalli stessi. Per ottenere un vero rumore bianco avremmo bisogno di commutare ad intervalli casuali, il che richiederebbe una routine piuttosto complessa per produrre un numero casuale e quindi controllare bit 4,A, provocando lunghe pause fra due successive commutazioni. Dobbiamo trovare pertanto un metodo rapido per produrre numeri casuali. Questo non è difficile come sembra poiché noi siamo interessati solo allo stato del bit 4,A; per i nostri scopi i byte della ROM sono sufficientemente casuali. Tenendo presente questo concetto, possiamo scrivere un programma che legga le varie celle della ROM, controlli lo stato del bit 4 e commuti alto o basso l'altoparlante.

Programma 7.7

```
org 42000 23760
23760 21 00 00     ld hl,0
L1
23763 7E           ld a,(hl)
23764 00           nop
23765 D3 FE        out (254),a
23767 00           nop
23768 00           nop
23769 D3 FE        out (254),a
23771 00           nop
23772 00           nop
23773 D3 FE        out (254),a
23775 00           nop
23776 23           inc hl
23777 7C           ld a,h
23778 FE 3C        cp 60
23780 20 ED        jr nz,L1
23782 C9           ret
```

È necessario focalizzare l'attenzione su alcune particolarità del programma 7.7.

1. Le tre istruzioni OUT (254),A e i NOP danno un certo ritardo. Potreste provare a diminuirle o ad aggiungerne altre e sentire l'effetto.
2. La lunghezza del suono è controllata dall'istruzione CP 60. Riducendo il termine di paragone a 40 (CP40), la lunghezza del suono verrebbe proporzionalmente ridotta. Superando il 60, si andrebbe ad attingere ad un'area della ROM con numeri non eccessivamente casuali. Provate a vedere che cosa succede.
3. Il colore della cornice viene anche temporaneamente modificato poiché i tre bit meno significativi variano anch'essi casualmente. Per mantenere costante il valore della cornice, avrete bisogno di sostituire i bit 0–2 dell'accumulatore con i bit 3–5 del valore contenuto nella variabile di sistema BORDER.

Se volete fare esperimenti producendo musica con questo metodo, ricordatevi che la routine di interrupt nella ROM viene chiamata 50 volte al secondo e falserebbe così la vostra frequenza; avrete bisogno di disabilitare gli interrupt (DI) prima della vostra routine e riabilitarli (EI) successivamente. Questo è quanto fa in effetti la routine nella ROM e spiega perché non potete eseguire un BREAK durante l'esecuzione di un comando BEEP. Lascio il resto alla vostra immaginazione e ingegnosità; provate ad alterare i valori contenuti nei registri HL e DE per produrre suoni ed effetti meravigliosi e strani. Per darvi una qualche idea dei suoni che

è possibile produrre usando il metodo appena descritto, il nastro contiene un programma SUONO che ha 5 differenti suoni generati dalla pressione dei tasti da 1 a 5. La cornice lampeggiante fornisce un'altra dimensione ai suoni 1 e 5 ed è molto facile da programmare. Un'ultima raccomandazione: mantenete gli effetti sonori più brevi possibile, poiché questi bloccano i movimenti grafici durante la loro esecuzione. La migliore posizione per i suoni è nei loop di ritardo usati per rallentare i movimenti.

Programma SUONO

```
  25 INK 2: PAPER 7: BORDER 4: CLS
  30 LET x=8:LET y=100:LET w=2:LET h=4:LET a$
="FERMA IL NASTRO"
  40 PAPER 8: RANDOMIZE USR 32393
  50 PRINT #0;AT 0,2;"PREMI UN TASTO PER INIZ
IARE"
  60 PAUSE 0
  65 CLS
  70 LET x=8: LET y=175: LET w=5: LET h=15: L
ET a$="SUONO!"
  80 INK 2: RANDOMIZE USR 32393
  90 PRINT AT 19,0;"Premi i tasti 1 - 5 per s
uonare"
 100 PRINT AT 21,3;"SPACE per tornare al BASI
C"
```

```
 110 RANDOMIZE USR 23760
 120 CLS
 130 LET x=5: LET y=100: LET h=5: LET w=1: LE
T a$="Per ricominciare digita GOTO 65"
 140 RANDOMIZE USR 32393
 150 STOP
9998 CLEAR 32334: LOAD ""CODE : GO TO 25


org 42000 23760
SUONO

START
23760 01 FE F7     ld bc,63486
23763 ED 78        in a,(c)
23765 F5           push af
23766 CB 47        bit 0,a
23768 CC 07 5D     call z,BEEP1
23771 F1           pop af
23772 F5           push af
23773 CB 4F        bit 1,a
23775 CC 29 5D     call z,BEEP2
23778 F1           pop af
23779 F5           push af
23780 CB 57        bit 2,a
23782 CC 44 5D     call z,BEEP3
23785 F1           pop af
23786 F5           push af
23787 CB 5F        bit 3,a

23789 CC 5F 5D     call z,BEEP4
23792 F1           pop af
23793 F5           push af
23794 CB 67        bit 4,a
23796 CC 7B 5D     call z,BEEP5
23799 F1           pop af
23800 3E 07        ld a,7
23802 CD 9B 22     call 8859
23805 01 FE 7F     ld bc,32766
23808 ED 78        in a,(c)
23810 CB 47        bit 0,a
23812 C8           ret z
23813 18 C9        jr START
BEEP1
23815 06 32        ld b,50
L1
23817 C5           push bc
23818 21 64 00     ld hl,100
L2
23821 11 0A 00     ld de,10
23824 E5           push hl
```

```
23825 7D          ld a,l
23826 E6 07       and 7
23828 CD 9B 22    call 8859
23831 CD B5 03    call 949
23834 E1          pop hl
23835 11 0A 00    ld de,10
23838 AF          xor a
23839 ED 52       sbc hl,de
23841 7C          ld a,h
23842 B5          or l
23843 20 E8       jr nz,L2
23845 C1          pop bc
23846 10 E1       djnz L1
23848 C9          ret
BEEP2
23849 21 00 00    ld hl,0
L3
23852 7E          ld a,(hl)
23853 00          nop
23854 D3 FE       out (254),a
23856 00          nop

23857 00          nop
23858 D3 FE       out (254),a
23860 00          nop
23861 00          nop
23862 D3 FE       out (254),a
23864 00          nop
23865 00          nop
23866 D3 FE       out (254),a
23868 00          nop
23869 23          inc hl
23870 7C          ld a,h
23871 FE 3C       cp 60
23873 20 E9       jr nz,L3
23875 C9          ret
BEEP3
23876 06 05       ld b,5
L5
23878 C5          push bc
23879 21 32 00    ld hl,50
L4
23882 11 0A 00    ld de,10

23885 E5          push hl
23886 CD B5 03    call 949
23889 E1          pop hl
23890 11 32 00    ld de,50
23893 19          add hl,de
23894 7C          ld a,h
```

```
23895 FE 04       cp 4
23897 20 EF       jr nz,L4
23899 C1          pop bc
23900 10 E8       djnz L5
23902 C9          ret
BEEP4
23903 06 0A       ld b,10
L6
23905 C5          push bc
23906 21 00 04    ld hl,1024
L7
23909 11 01 00    ld de,1
23912 E5          push hl
23913 CD B5 03    call 949
23916 E1          pop hl

23917 AF          xor a
23918 11 10 00    ld de,16
23921 ED 52       sbc hl,de
23923 7C          ld a,h
23924 B5          or l
23925 20 EE       jr nz,L7
23927 C1          pop bc
23928 10 E7       djnz L6
23930 C9          ret
BEEP5
23931 21 01 00    ld hl,1
L8
23934 11 05 00    ld de,5
23937 E5          push hl
23938 CD B5 03    call 949
23941 E1          pop hl
23942 23          inc hl
23943 7C          ld a,h
23944 FE 02       cp 2
23946 20 F2       jr nz,L8
23948 06 FA       ld b,250

L9
23950 C5          push bc
23951 78          ld a,b
23952 E6 06       and 6
23954 CD 9B 22    call 8859
23957 21 90 01    ld hl,400
23960 11 01 00    ld de,1
23963 CD B5 03    call 949
23966 C1          pop bc
23967 10 ED       djnz L9
23969 C9          ret
```

Capitolo

ATTRIBUTE, SCREEN$ e POINT

8

Se avete già provato a programmare giochi in BASIC saprete che, prima o poi, viene un momento in cui è necessario conoscere il carattere presente in una certa posizione dello schermo. Questo è utile, ad esempio, per controllare che un missile abbia colpito il bersaglio o che voi siate felicemente atterrati sulla Luna. Tre sono i metodi messi a disposizione dal BASIC:
ATTRIBUTE, che controlla lo stato di INK, PAPER, FLASH, BRIGHT sulla posizione di un carattere e restituisce un numero da 0 a 255.
SCREEN$, che verifica il carattere presente in una particolare posizione (ma restituisce un risultato diverso da una stringa nulla solo se il carattere presente ha codice compreso tra 32 e 127).
POINT, che controlla lo stato di un pixel dello schermo:
0 = spento, 1 = acceso.
Tutti e tre i metodi appena presentati sono disponibili in codice macchina tramite chiamate alle relative routine nella ROM.

ATTRIBUTE (linea, colonna)

La routine in codice macchina (call 9603) richiede il numero di linea nel registro C ed il numero di colonna nel registro B. Il valore di ATTRIBUTE sarà posto dalla routine in cima allo stack di calcolo. Esso è così composto:

ATTRIBUTE = 128*FLASH+64*BRIGHT+8*PAPER+INK

Come potete vedere dalla formula, il campo di variazione di ATTRIBUTE è da 0 a 255 e può quindi essere facilmente prelevato dallo stack e posto nel registro A tramite la routine CALC VALUE TO A (call 11733); il valore così ottenuto può essere controllato tramite un'istruzione CP n.
Il programma 8.1 dimostra il metodo usato per ottenere PRINT AT 0, 20; PAPER 1; INK 7; FLASH 1; CHR$ 144 (che è stato ridefinito come un omino). Il programma controlla ogni posizione dello schermo, mostrata da un segno >, finché trova in ATTR il valore 143 e quindi ritorna al BASIC.

Programma 8.1

```
org 42000 23760
CONTROLLA GLI 'ATTRIBUTE'

PREDISPOSIZIONE UDG

23760 ED 5B 7B 5C ld de,(23675)
23764 21 42 5D    ld hl,DATA3
23767 01 08 00    ld bc,8
23770 ED B0       ldir

CLS

23772 3E 02       ld a,2
23774 CD 01 16    call 5633
23777 CD 6B 0D    call 3435
23780 3E 02       ld a,2
23782 CD 01 16    call 5633

PRINT UDG

23785 11 2E 5D    ld de,DATA1
23788 01 10 00    ld bc,16
23791 CD 3C 20    call 8252

PONE OVER 1

23794 FD 36 57 03 ld (iy+87),3

POSIZIONE INIZIALE 0,0

23798 AF          xor a
23799 32 40 5D    ld (COL),a

PRINT >

L1
23802 11 3E 5D    ld de,DATA2
```

```
23805 01 04 00     ld bc,4
23808 CD 3C 20     call 8252

RITARDO

23811 21 FF FF     ld hl,65535
L2
23814 2B           dec hl
23815 7C           ld a,h
23816 B5           or l
23817 20 FB        jr nz,L2

RIMUOVE >

23819 11 3E 5D     ld de,DATA2
23822 01 04 00     ld bc,4
23825 CD 3C 20     call 8252

INCREMENTA COLONNA

23828 3A 40 5D     ld a,(COL)
23831 3C           inc a
23832 32 40 5D     ld (COL),a

PONE COLONNA/LINEA IN BC

23835 ED 4B 3F 5D ld bc,(LINE)

CALL ATTR. &
PONE IL VALORE IN A

23839 CD 83 25     call 9603
23842 CD D5 2D     call 11733

CONTROLLA SE E' 143

23845 FE 8F        cp 143

PASSA ALLA POSIZIONE SUCCESSIVA

23847 20 D1        jr nz,L1

RITORNA A PORRE OVER 0

23849 FD 36 57 00 ld (iy+87),0
23853 C9           ret

PRINT UDG DATA
```

```
DATA1
defb 22 0 20 17 1 16 7 18 1
defb 144 18 0 17 7 16 0

PRINT > DATA

DATA2
defb 22
LINE
defb 0
COL
defb 0 62

UDG DATA

DATA3
defb 24 153 126 153 24 36 36 102
```

Il programma 8.1 rappresenta un missile che si muove lentamente verso un uomo; passa al BASIC quando l'uomo viene colpito. Potete ovviamente avere, al posto dell'istruzione RETURN, una routine di "uomo colpito" (con effetti sonori). Questo metodo di controllo della posizione dello schermo per i bersagli è ottimo, ma è un po' limitante poiché tutti i vostri bersagli devono avere lo stesso valore di ATTRIBUTE.

SCREEN$ (linea, colonna)

Analogamente alla routine ATTR, il numero di linee è contenuto nel registro C ed il numero di colonne nel registro B prima di eseguire call 9528. La routine piazza i parametri della stringa in cima allo stack di calcolo sotto forma di 5 byte. Nuovo è l'uso dello stack di calcolo per immagazzinare i parametri della stringa in quanto finora è stato sfruttato solo per memorizzare numeri a 5 byte. Basti dire che esistono i literal di calcolo per la manipolazione delle stringhe ma, invece di mettere fisicamente la stringa sullo stack, immagazziniamo i suoi parametri in una forma a 5 byte. Per far ciò, la coppia di registri BC contiene la lunghezza della stringa e la coppia di registri DE il suo indirizzo iniziale. Per semplici stringhe, il registro A contiene 0. Chiamando la routine 10929 immagazzineremo nel corretto ordine i parametri sullo stack. Analogamente, la routine 11249 preleverà dallo stack l'ultimo valore e lo piazzerà nei registri A, B, C, D, ed E; BC conterrà nuovamente la lunghezza della stringa e DE l'indirizzo iniziale.
Il programma 8.2 mostra l'uso delle routine per visualizzare il carattere @ nella posizione di schermo 0,20; l'esecuzione del programma terminerà

con due @ in 0,20 e 0,21. Invece di usare PRINT STRING (call 8252) avremmo potuto usare LD A, (DE) per controllare il carattere attualmente alla posizione 0,20 e proseguire quindi conseguentemente.
Comunque, la routine BASIC SCREEN$ limita la propria efficacia ai caratteri il cui codice è compreso fra 32 e 127 inclusi; qualunque altro carattere restituirà una stringa vuota. Questo metodo non va bene per trovare gli UDG dal BASIC, il che è un grosso handicap nei programmi di giochi. In codice macchina non esiste problema, essendoci metodi veloci per ingannare la ROM e far rientrare gli UDG nel set di caratteri. La variabile di sistema CHARS (23606/7) generalmente contiene l'indirizzo iniziale del set di caratteri −256, cioè 15360, ed è usata dalla routine SCREEN$ per trovare il carattere. Potremmo cambiare questa variabile per far sì che questa punti all'inizio degli UDG — 256 prima di chiamare la routine SCREEN$ (in effetti, facendo sì che UDG A assuma il codice 32, UDG B il codice 33,...), resettandola in seguito al suo valore normale. In questo modo possiamo esaminare anche gli UDG per trovare il carattere desiderato.

Programma 8.2

```
org 42000 23760
SCREEN$ (0,20)

CLS

23760 3E 02       ld a,2
23762 CD 01 16    call 5633
23765 CD 6B 0D    call 3435
23768 3E 02       ld a,2
23770 CD 01 16    call 5633

PRINT AT 0,20 @

23773 3E 16       ld a,22
23775 D7          rst 16
23776 3E 00       ld a,0
23778 D7          rst 16
23779 3E 14       ld a,20
23781 D7          rst 16
23782 3E 40       ld a,64

23784 D7          rst 16

CALL SCREEN$

23785 06 14       ld b,20
23787 0E 00       ld c,0
23789 CD 38 25    call 9528
```

```
CALL 'VALUE TO A,B,C,D,E

23792 CD F1 2B     call 11249

CALL PRINT STRING

23795 CD 3C 20     call 8252
23798 C9           ret
```

Il programma 8.3 è analogo al programma 8.1; questa volta, però prima di tornare al BASIC, verifica la presenza di CHR$ 144.

Programma 8.3

```
org 42000 23836
SCREEN$ PER UDG


da 23760 a 23835
IDENTICO AL PROGRAMMA 8.1

PONE I CARATTERI A UDG - 256

23836 2A 7B 5C     ld hl,(23675)
23839 25           dec h
23840 22 36 5C     ld (23606),hl

CONTROLLA SCREEN$

23843 CD 38 25     call 9528
23846 CD F1 2B     call 11249

CONTROLLA SE CHR$ 32
cioe' CHR$ 144 (UDG 'A')


23849 1A           ld a,(de)
23850 FE 20        cp 32

RESETTA I CARATTERI ALLA
SITUAZIONE NORMALE

23852 21 00 3C     ld hl,15360
23855 22 36 5C     ld (23606),hl

SEGUITO DEL PROGRAMMA
COME NEL PROGRAMMA 8.1

DA 23847 ALLA FINE DI DATA3
```

È ovviamente possibile copiare il set di caratteri nella RAM e far sì che CHARS punti permanentemente all'inizio del nuovo set −256. In questo modo potete trascurare gli UDG e ridefinire qualunque carattere nel nuovo set (lasciando invariati l'alfabeto ed i numeri per non alterare l'eventuale testo). Questo metodo vi permetterà di ridefinire circa 50 caratteri, purché nei testi vi limitiate all'uso di caratteri maiuscoli e di un minimo di segni di interpunzione. La routine SCREEN$ riconoscerà questi caratteri così come appaiono nel nuovo set.
Usando lo stack per la manipolazione delle stringhe, è importante notare che viene occupato spazio di lavoro per la memorizzazione della stringa stessa e/o del risultato dell'elaborazione. Trascurando di ripulire regolarmente quest'area di lavoro, potrebbe apparirvi un messaggio di OUT OF MEMORY senza alcuna apparente ragione o, peggio ancora, potreste, con la vostra area di lavoro, invadere la zona riservata alla routine in codice macchina, causando il blocco del sistema. È pertanto consigliabile ripulire l'area di lavoro dopo ogni chiamata alla routine SCREEN$. Una routine (call 5823) ripulirà l'area di lavoro e lo stack di calcolo. Pertanto il programma 8.3 dovrebbe chiamare la routine 5823 dopo l'istruzione LD A,(DE); ricordatevi di salvare, prima della chiamata a questa routine, il valore in A.

POINT

Il comando BASIC POINT x, y restituisce il valore 1 se il pixel identificato dalle coordinate x, y è acceso (PLOT), o 0 se è spento (PLOT OVER). La routine in codice macchina (call 8910) richiede la coordinata x nel registro C e la coordinata y nel registro B. Come le routine precedentemente descritte, anche questa depone il risultato sullo stack di calcolo e, essendo questo 0 o 1, ci è possibile prelevarlo dallo stack e confrontarlo col valore richiesto. Il programma 8.4 mostra come sfruttare questo metodo per controllare la posizione di schermo 0,175, cioè l'angolo superiore sinistro. La routine prima visualizza un quadretto nero in 0,0, quindi ritorna al BASIC come POINT 0,175 = 1.

Programma 8.4.

```
org 42000 23760
POINT x,y

23760 3E 02        ld a,2
```

```
23762 CD 01 16     call 5633
23765 CD 6B 0D     call 3435
23768 3E 02        ld a,2
23770 CD 01 16     call 5633
23773 3E 8F        ld a,143
23775 D7           rst 16
23776 0E 00        ld c,0
23778 06 AF        ld b,175
23780 CD CE 22     call 8910
23783 CD D5 2D     call 11733
23786 FE 01        cp 1
23788 C8           ret z
```

Capitolo

La stampante 9

Non tutti i programmi usano la stampante ma può talvolta essere utile avere una copia dello schermo. Per questo motivo esamineremo i tre comandi BASIC COPY, LPRINT e LLIST e i loro equivalenti in codice macchina.

COPY

Normalmente, la routine COPY passa il contenuto delle prime 22 linee dello schermo direttamente alla stampante. In codice macchina possiamo migliorare la versatilità di questa routine specificando quante linee di schermo vogliamo siano copiate e la linea iniziale; potremo così copiare, ad esempio, anche solo tre linee, partendo dalla settima.
Il programma 9.1 mostra come copiare tutte le 24 linee, caratteristica questa, utile in alcuni programmi di giochi che visualizzano il punteggio nelle linee inferiori, non altrimenti considerate dal normale comando COPY. Come potete vedere, prima di chiamare la routine 3762, la coppia di registri HL contiene l'indirizzo della linea di partenza, nel nostro esempio è la linea 1, e il registro B contiene il numero di linee da copiare moltiplicato 8 ($24 \times 8 = 192$).

Programma 9.1

```
org 42000 23760
COPIA SULLA STAMPANTE
 n LINEE DELLO SCHERMO
```

```
HL=INDIRIZZO INIZIO PRIMA LINEA
 B=NUMERO LINEE x 8

ESEMPIO 24 LINEE

23760 21 00 40     ld hl,16384
23763 06 C0        ld b,192
23765 F3           di
23766 CD B2 0E     call 3762
23769 C9           ret
```

LPRINT

Questa istruzione è analoga a PRINT, con la sola differenza che, ovviamente, i caratteri di controllo non avranno alcun effetto e che è necessario aprire un differente canale prima di eseguire l'istruzione RST 16 per mandare l'output alla stampante invece che allo schermo. Volendo pertanto ottenere LPRINT A è prima necessario fare OPEN CHANNEL 3, LD A, CODE A e quindi usare RST 16. Il programma 9.2 mostra come sfruttare il metodo appena descritto per stampare una stringa di caratteri. Per ottenere una tabulazione sulla vostra stampa, il numero di TAB va inserito (POKE) nella variabile di sistema PRCC 23680 prima di eseguire l'istruzione RST 16.

Programma 9.2

```
org 42000 23760
LPRINT

APRE IL CANALE 3

23760 3E 03        ld a,3
23762 CD 01 16     call 5633
23765 11 DF 5C     ld de,DATA
23768 01 11 00     ld bc,17

LPRINT STRING

23771 CD 3C 20     call 8252
23774 C9           ret
DATA
defs Questo e' il programma 2
```

LLIST

Ho incluso LLIST, anche se non mi viene in mente alcuna applicazione che possa richiedere un listato nel mezzo del programma, perché è uno dei comandi della stampante ed è molto facile da ottenere. Tutto quello che dovete fare è call 6133. Il programma 9.3 dimostra questo sistema assemblando il programma BASIC e quindi listandolo sulla stampante.

Programma 9.3

```
org 42000 23760
LLIST

23760 CD F5 17     call 6133
23763 C9           ret
```

Codice generato dal programma 9.3

```
  20 REM go
  30 REM org 42000 23760
  40 REM !CAPITOLO 9;!PROGRAMMA 3;!
  50 REM ! LLIST ;!
  60 REM call 6133
  70 REM ret
9000 REM finish
9010 RANDOMIZE USR 58000
9020 COPY
9030 LPRINT
9040 RANDOMIZE USR 23760
```

Codice Sorgente del Programma 9.3

Capitolo

Conversione dei programmi

10

Mi complimento con tutti coloro che sono riusciti a raggiungere questo capitolo avendo compreso a fondo i precedenti; se invece vi resta qualche dubbio sulle routine presentate, vi consiglio di chiarirvelo prima di continuare la lettura.
In questo capitolo cercherò di dimostrare come produrre un programma di giochi totalmente in codice macchina, usando come guida un programma BASIC.
Avrete senza dubbio un metodo personale di scrivere un programma BASIC. Forse partite dall'idea e la sviluppate sullo Spectrum aggiungendo routine via via che vi vengono in mente, ottenendo così un programma funzionante ma completamente privo di struttura, comprensibile solo da voi. Questo metodo può essere soddisfacente ma può anche darsi che qualche settimana dopo, provando a modificare il programma, vi accorgiate di non essere più in grado, nemmeno voi, di comprenderlo. Penso che il metodo migliore consista nel realizzare un diagramma di flusso partendo dall'idea di base, aggiungendo routine via via che queste si rivelano necessarie, per finire così con un diagramma di flusso totalmente strutturato, da cui ricavare il programma BASIC da rifinire successivamente con suoni, colori e UDG. Questo metodo, inoltre, produce un programma facilmente convertibile in codice macchina. In questo capitolo convertiremo il programma CROSS. L'idea di base da cui è stato ricavato il diagramma di flusso, consiste nel guidare un omino dotato di numerose "vite" (come i gatti), attraverso una serie di ostacoli, fino ad un rifugio sicuro. La routine principale provvederà ad inizializzare le variabili e a visualizzare le istruzioni ed il terreno di gioco, quindi si sfrutterà un loop per controllare i movimenti dell'omino e la sua nuova posizione sul-

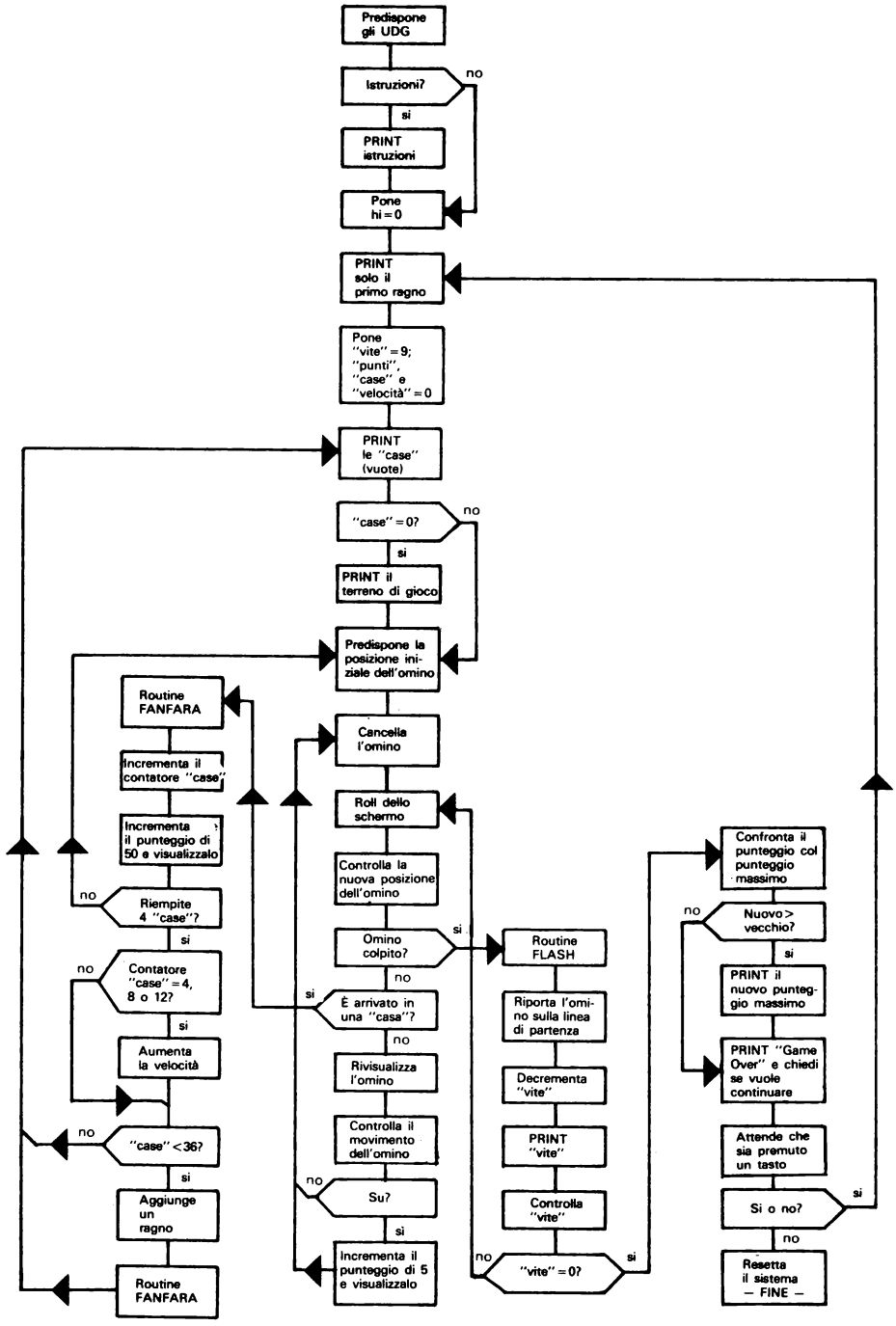


Figura 10.1 Diagramma di flusso del programma CROSS.

lo schermo. Esistono due uscite da questo loop principale:

1. Se l'uomo viene colpito, viene chiamata la routine HIT e si controllano le vite rimanenti; se queste sono finite, il gioco termina, altrimenti si torna nel loop principale.
2. La routine HOME che controlla il numero di case riempite, aggiunge ragni ed aumenta la velocità prima di ritornare al loop principale.

Da questo diagramma di flusso è stata ottenuta la versione BASIC del programma CROSS. Se confrontate il programma stesso con il diagramma di flusso, sarete facilmente in grado di distinguere le varie routine. Il programma, anche se molto semplice, viene enormemente migliorato dal pieno uso degli UDG, dei colori e degli effetti sonori. Sembra strano aver usato, per ottenere il roll a destra o a sinistra delle linee dello schermo un pixel per volta, una routine in codice macchina, ma la sua implementazione in BASIC sarebbe stata enormemente lenta.
Il codice macchina viene posto (tramite dei POKE) nelle corrette locazioni di memoria del programma BASIC. Si sarebbe anche potuto usare una istruzione di tipo LOAD code XX, n per caricare il codice macchina direttamente dal nastro.
Prima di proseguire la lettura, assicuratevi di aver completamente compreso il funzionamento del programma, poiché noi ora lo esamineremo linea per linea convertendolo in codice macchina.

Programma CROSS versione BASIC

```
   20 CLEAR 32243: GO TO 40
   25 BEEP .01,b-a
   30 PRINT OVER 1; PAPER 8; INK
8;AT a,y2;"X": RETURN
   40 PRINT AT 11,7;"Attendere pr
ego"
   50 FOR a=32244 TO 32494
   60 READ b: POKE a,b: NEXT a
   70 DATA 14,8,229,17,31,0,25,12
6,237,82,31,6,32,126,31,119,35,1
6,250,225,36,13,32,234,201
   80 DATA 14,8,175,229,17,31,0,2
37,82,126,25,23,6,32,126,23,119,
43,16,250,225,36,13,32,233,201
   90 DATA 33,95,64,205,13,126,33
,128,64,205,244,125,33,128,64,20
5,244,125,33,223,64,205,13,126,3
3,0,72,205,244,125,33,0,72,205,2
44,125,33,0,72,205,244,125
  100 DATA 58,121,92,0,0,0,0,0,0,
0,230,2,40,20,33,64,72,205,244,1
25,33,64,72,205,244,125,33,64,72
,205,244,125,24,18,33,95,72,205,
13,126,33,95,72,205,13,126,33,95
,72,205,13,126
  110 DATA 33,128,72,205,244,125,
33,192,72,205,244,125,33,192,72,
205,244,125,33,31,80,205,13,126,
33,31,80,205,13,126,33,95,80,205
,13,126,201
```

```
 120 DATA 33,128,72,205,244,125,
33,192,72,205,244,125,33,31,80,2
05,13,126,33,95,80,205,13,126,20
1
 130 DATA 33,95,64,205,13,126,33
,128,64,205,244,125,33,0,72,205,
244,125,201
 140 DATA 33,95,64,205,13,126,33
,223,64,205,13,126,33,128,72,205
,244,125,33,192,72,205,244,125,2
01
 150 LET a=PEEK 23675+256*PEEK 2
3676
 160 FOR b=a TO a+167
 170 READ c: POKE b,c: NEXT b
 180 DATA 15,18,34,127,255,255,4
0,16,128,64,32,254,254,255,40,16
 190 DATA 127,127,127,127,127,25
5,21,8,254,254,254,254,255,255,6
4,128
 200 DATA 0,248,196,196,254,254,
40,16,24,24,36,126,60,90,165,66
 210 DATA 56,40,146,124,56,56,40
,108,1,2,4,127,127,255,20,8
 220 DATA 240,72,68,254,255,255,
20,8,0,31,35,35,127,127,20,8
 230 DATA 127,127,127,127,255,25
5,2,1,254,254,254,254,254,255,16
8,16
 240 DATA 16,41,199,0,36,0,0,0,0
,68,255,68,68,255,68,0
 250 DATA 0,34,85,143,151,163,16
0,0,0,68,170,241,233,197,5,0
 260 DATA 16,16,16,254,63,31,15,
7,0,0,0,0,30,255,255,255
 270 DATA 96,124,84,120,127,255,
254,252,0,0,3,2,15,63,255,0,6,12
,152,240,224,85,255,0
 280 PRINT AT 11,5;"Vuoi le istr
uzioni ?";AT 13,13;"(s)i";AT 15,
13;"(n)o"
 290 PAUSE 0: IF INKEY$="s" THEN
 GO TO 300
 295 GO TO 400
 300 CLS : PRINT AT 0,11;"OGGETT
O";"Guidare un * attraverso str
ada,evitando                  *
";"Un  pattuglia l'isola
 centrale"
 310 PRINT '''"Dovete riempire le
 4 CASE,cioe'ivuoti sul bordo su
periore ## ##"
 320 PRINT '"Una volta riempite,
le 4 CASE si svuoteranno,la velo
cita' salira'e verra' aggiunto u
n altro  "
 330 PRINT '''AT 18,9;"Premi un
tasto": PAUSE 0
 370 CLS : PRINT AT 7,11;"COMAND
I"
 375 PRINT '"          ↑
    <  >"
 380 PRINT FLASH 1;AT 11,6;"1";
FLASH 0;" 2 3 4 5 6 7 8 "; FLASH
 1;"9"; FLASH 0;" "; FLASH 1;"0"
 390 PRINT AT 18,3;"Per giocare
premi un tasto": PAUSE 0
 400 BRIGHT 1: PAPER 5: BORDER 5
: CLS
 410 LET hi=0
 420 PRINT PAPER 4;AT 10,0;"
                 "
 430 LET lives=9: LET score=0: L
ET home=0
 440 POKE 32425,201: POKE 32450,
```

```
201: POKE 32469,201
 450 PRINT AT 0,0; PAPER 4;"######
";  PAPER 7;"  "; PAPER 4;"##########"
;  PAPER 7;"  "; PAPER 4;"##########"
;  PAPER 7;"  "; PAPER 4;"##########"
;  PAPER 7;"  "; PAPER 4;"######"
 455 IF home<>0 THEN GO TO 660
 460 PRINT PAPER 4; INK 5;"
                                    "
 470 PRINT "
    "
 480 PRINT INK 7;"
"
 490 PRINT INK 2;"
    "
 500 PRINT INK 7;"
    "
 510 PRINT INK 1;"
    "
 520 PRINT INK 7;"
    "
 530 PRINT "
    "
 540 PRINT PAPER 4;"##################
##############################"
 550 PRINT PAPER 0; INK 7;AT 11,
0;"##################################
#####"
 560 PRINT PAPER 0; INK 3;"
    "
 570 PRINT PAPER 0; INK 7;"------
--------------------------"
 580 PRINT PAPER 0; INK 5;"
    "
 590 PRINT PAPER 0; INK 7;"======
========================="
 600 PRINT PAPER 0; INK 4;"
    "
 610 PRINT PAPER 0; INK 7;"------
--------------------------"
 620 PRINT PAPER 0; INK 6;"
    "
 630 PRINT PAPER 4;"##################
##############################"
 640 PRINT PAPER 4;"
    "
 650 PRINT PAPER 1; INK 7;" SCOR
E  ";AT 21,11;" MEN "; PAPER 5; I
NK 0;lives; PAPER 1; INK 7;" HI-
SCORE "
 660 LET x1=20: LET y1=16: LET x
2=x1: LET y2=y1
 670 PRINT PAPER 8; INK 8;AT x1,
y1;" "
 680 RANDOMIZE USR 32295
 690 IF SCREEN$ (x2,y2)=" " THEN
 GO TO 680
 700 LET a=x2: FOR b=25 TO 35: G
O SUB 25: GO SUB 25: NEXT b
 730 FOR a=x2 TO 20 STEP 2: GO S
UB 25: GO SUB 25: NEXT a
 740 LET lives=lives-1: PRINT AT
 21,16;lives
 750 LET x2=20
 760 IF lives<>0 THEN GO TO 680
 770 IF hi>score THEN GO TO 790
 780 LET hi=score: PRINT AT 21,2
7;hi
 790 PRINT FLASH 1; PAPER 7;AT 1
2,0;"
                GAME  OVER
    "
 800 PRINT AT 14,0;"Vuoi giocare
 ancora ? (s)i (n)o "
 840 IF INKEY$="n" THEN RANDOMIZ
E USR 0
```

```
 850 IF INKEY$<>"s" THEN GO TO 8
40
 860 PRINT PAPER 5;AT 21,7;"
": GO TO 415
 880 IF x2<>0 THEN GO TO 1050
 890 PRINT PAPER 8; INK 8;AT x1,
y1," ": PRINT AT x2,y2;"☺"
 900 RESTORE 920
 910 FOR a=1 TO 8: READ b,c: BEE
P b,c: NEXT a
 920 DATA .1,11,.1,11,.8,16,.05,
11,.05,16,.05,11,.05,16,1,20
 930 LET home=home+1: LET score=
score+50: PRINT AT 21,7;score
 950 IF home/4<>INT (home/4) THE
N GO TO 660
 960 IF home=4 THEN POKE 32425,0
 970 IF home=8 THEN POKE 32450,0
 980 IF home=12 THEN POKE 32469,
0
 985 IF home>36 THEN GO TO 450
 990 LET a=RND*30
1000 LET a=a+1
1005 IF a>31 THEN LET a=0
1010 IF SCREEN$ (10,a)="" THEN G
O TO 1000
1020 IF SCREEN$ (10,a+1)="" THEN
 GO TO 1000
1030 PRINT PAPER 4;AT 10,a;"🐍"
1035 RESTORE 920: FOR a=1 TO 8:
READ b,c: BEEP b,c: NEXT a
1040 GO TO 450
1050 PRINT PAPER 8; INK 8;AT x2,
y2;"☺"
1060 LET x1=x2: LET y1=y2
1070 IF INKEY$<>"1" THEN GO TO 1
100
1080 BEEP .001,33
1090 LET x2=x2-2: LET score=scor
e+5: PRINT AT 21,7;score
1100 LET y2=y2+(INKEY$="0" AND y
2<>31)-(INKEY$="9" AND y2<>0)
1110 GO TO 670
```

CARATTERI GRAFICI

AB = 

CDE = 

F = 

G = ☺

HI = 

JKL = 

M = ☆

N = #

OP = 

QRS = 

TU = 

Codice macchima per il programma CROSS versione BASIC

```
org 42000 32244
CAPITOLO 10
PROGRAMMA >CROSS<

[ BASIC ] CODICE MACCHINA

ROLL DELLA RIGA VERSO DESTRA

RIGHT
32244 0E 08        ld c,8
L1
32246 E5           push hl
32247 11 1F 00     ld de,31
32250 19           add hl,de
32251 7E           ld a,(hl)
32252 ED 52        sbc hl,de
32254 1F           rra
32255 06 20        ld b,32
L2

32257 7E           ld a,(hl)
32258 1F           rra
32259 77           ld (hl),a
32260 23           inc hl
32261 10 FA        djnz L2
32263 E1           pop hl
32264 24           inc h
32265 0D           dec c
32266 20 EA        jr nz,L1
32268 C9           ret

ROLL DELLA RIGA VERSO SINISTRA

LEFT
32269 0E 08        ld c,8
L3
32271 AF           xor a
32272 E5           push hl
32273 11 1F 00     ld de,31
32276 ED 52        sbc hl,de
32278 7E           ld a,(hl)

32279 19           add hl,de
32280 17           rla
32281 06 20        ld b,32
L4
32283 7E           ld a,(hl)
32284 17           rla
```

```
32285 77          ld (hl),a
32286 2B          dec hl
32287 10 FA       djnz L4
32289 E1          pop hl
32290 24          inc h
32291 0D          dec c
32292 20 E9       jr nz,L3
32294 C9          ret

ROUTINE DI MOTO


HL contiene :
Inizio riga per ROLL a DESTRA
Fine riga per ROLL a SINISTRA


SPEED 0

32295 21 5F 40    ld hl,16479
32298 CD 0D 7E    call LEFT
32301 21 80 40    ld hl,16512
32304 CD F4 7D    call RIGHT
32307 21 80 40    ld hl,16512
32310 CD F4 7D    call RIGHT
32313 21 DF 40    ld hl,16607
32316 CD 0D 7E    call LEFT
32319 21 00 48    ld hl,18432
32322 CD F4 7D    call RIGHT
32325 21 00 48    ld hl,18432
32328 CD F4 7D    call RIGHT
32331 21 00 48    ld hl,18432
32334 CD F4 7D    call RIGHT

Muove la riga del RAGNO
Usa il contatore FRAMES


32337 3A 79 5C    ld a,(23673)
32340 00          nop
32341 00          nop
32342 00          nop
32343 00          nop
32344 00          nop
32345 00          nop
32346 00          nop
32347 E6 02       and 2
32349 28 14       jr z,L5
32351 21 40 48    ld hl,18496
32354 CD F4 7D    call RIGHT
```

```
32357 21 40 48    ld hl,18496
32360 CD F4 7D    call RIGHT
32363 21 40 48    ld hl,18496
32366 CD F4 7D    call RIGHT
32369 18 12       jr L6
L5
32371 21 5F 48    ld hl,18527
32374 CD 0D 7E    call LEFT
32377 21 5F 48    ld hl,18527

32380 CD 0D 7E    call LEFT
32383 21 5F 48    ld hl,18527
32386 CD 0D 7E    call LEFT
L6
32389 21 80 48    ld hl,18560
32392 CD F4 7D    call RIGHT
32395 21 C0 48    ld hl,18624
32398 CD F4 7D    call RIGHT
32401 21 C0 48    ld hl,18624
32404 CD F4 7D    call RIGHT
32407 21 1F 50    ld hl,20511
32410 CD 0D 7E    call LEFT
32413 21 1F 50    ld hl,20511
32416 CD 0D 7E    call LEFT
32419 21 5F 50    ld hl,20575
32422 CD 0D 7E    call LEFT
32425 C9          ret

SPEED 1

32426 21 80 48    ld hl,18560

32429 CD F4 7D    call RIGHT
32432 21 C0 48    ld hl,18624
32435 CD F4 7D    call RIGHT
32438 21 1F 50    ld hl,20511
32441 CD 0D 7E    call LEFT
32444 21 5F 50    ld hl,20575
32447 CD 0D 7E    call LEFT
32450 C9          ret

SPEED 2

32451 21 5F 40    ld hl,16479
32454 CD 0D 7E    call LEFT
32457 21 80 40    ld hl,16512
32460 CD F4 7D    call RIGHT
32463 21 00 48    ld hl,18432
32466 CD F4 7D    call RIGHT
32469 C9          ret
```

```
SPEED 3


32470 21 5F 40     ld hl,16479
32473 CD 0D 7E     call LEFT
32476 21 DF 40     ld hl,16607
32479 CD 0D 7E     call LEFT
32482 21 80 48     ld hl,18560
32485 CD F4 7D     call RIGHT
32488 21 C0 48     ld hl,18624
32491 CD F4 7D     call RIGHT
32494 C9           ret
```

CONVERSIONE IN CODICE MACCHINA

LINEA 20

Non è necessaria nella versione in codice macchina, perché è inutile abbassare la RAMTOP per proteggere il codice macchina.

LINEE 25 E 30

È una subroutine usata dalla routine HIT, con cui farà tutt'uno nella versione in codice macchina. La subroutine è posta all'inizio del programma BASIC per ridurre il tempo necessario allo Spectrum per reperirla. Alla chiamata delle subroutine, lo Spectrum inizia a ricercarle dall'inizio del programma BASIC, scandendo i vari numeri di linea, fino a trovare quello richiesto. Osservate che, se la routine avesse avuto un numero di linea più alto, sarebbe stato necessario più tempo per ritrovarla, con conseguente aumento del tempo di esecuzione delle linee da 700 a 730, che chiamano questa routine almeno 22 volte.

LINEA 40

In BASIC sono necessari vari secondi per rilocare opportunamente il codice macchina e predisporre gli UDG; è quindi opportuno visualizzare il messaggio durante questo periodo d'attesa. In codice macchina, tuttavia, quest'operazione è praticamente istantanea e il messaggio è pertanto inutile.

LINEE 50–140

Questa è la routine per rilocare (POKE) la routine ROLL in codice macchina negli indirizzi di memoria da 32244 a 32494; anche questo non è ri-

chiesto dal codice macchina, dato che è automaticamente piazzato alle opportune locazioni insieme al resto del programma in codice macchina.

LINEE 150–270 PREDISPOSIZIONE UDG

Questa è la prima routine che sarà necessario convertire. Avremmo ovviamente potuto usare SAVE UDG CODE USR a, 168: LOAD UDG CODE a, 168, ma ciò avrebbe richiesto la registrazione al successivo caricamento di due routine in codice macchina, cioè il gioco e gli UDG. Per far lavorare bene il programma anche nello Spectrum 16K, lo faremo iniziare all'indirizzo 24001 (5DClh). Analogamente a quanto già fatto nel BASIC, memorizzeremo ogni byte degli UDG sotto forma di DATA, che leggeremo trasferendo ogni numero al suo posto nella zona di memoria riservata agli UDG. Il metodo più semplice sfrutta l'istruzione LDIR. La coppia di registri DE contiene l'indirizzo iniziale degli UDG, come prelevato dalla variabile di sistema 23675 (5C7Bh) e la coppia di registri HL l'indirizzo iniziale dei DATA. I dati sono immagazzinati nelle locazioni da 5DClh a 5E68h e la routine in codice macchina da 24169 (5E69h) a 24180 (5E73h). Da notare che la routine viene chiamata usando RANDOMIZE USR 24169 e non USR 24001, che è l'inizio dei dati.

LINEA 280

È un semplice comando di visualizzazione, facilmente convertibile in codice macchina usando la routine ROM 8252 (203Ch), come presentato nel Capitolo 1. Quindi DATA = 24183 (5E77h) a 24219 (5E9Bh). Questi dati vengono saltati tramite l'istruzione JR 5E9Ch all'indirizzo 5E75/6h. La routine PRINT DATA = 5E9Ch (24220) a 5EA9h.

LINEE 290 E 295 CODICE MACCHINA 5EAA–5EB8

In questa routine attendiamo la pressione di un tasto e controlliamo se sia stato premuto quello corrispondente alla lettera "s" (non è necessario il comando HALT). Se il tasto premuto non è "s", tralasceremo le istruzioni saltando (NZ) a 24714 (608Ah).

LINEA 300: 1 CLS CODICE MACCHINA 5EB9–5EC0

Questa routine è ampiamente discussa nel Capitolo 1.

LINEE 300: 2–330

Usiamo ancora la forma di memorizzazione come DATA e le routine di visualizzazione di stringa. Il primo blocco di DATA va da 5ECA a 5F72, e la

routine di visualizzazione da 5F75 a 5F85. Il secondo blocco di DATA è memorizzato da 5F86 a 5FF7, con la relativa routine di visualizzazione da 5FF8 a 6000. I dati avrebbero potuto essere trattati da un'unica routine di PRINT DATA. La routine di PAUSE 0 va da 6001 a 600A; da notare che bit 5, (IY+1) deve essere resettato dalla precedente routine di PAUSE 0. Provate a vedere cosa succederebbe rimuovendo l'istruzione RES 5, (IY+1).

LINEA 370: 1 ROUTINE CLS 600C–6010

Sfrutta il metodo alternativo di ripulire l'intero schermo ponendo (POKE) zeri in tutti gli indirizzi della memoria video (senza toccare gli attributi).

LINEE 370: 2–390

DATA per la routine di visualizzazione = 6014–6075
Routine PRINT DATA = 6076–607E
Viene ulteriormente seguita dalla routine PAUSE 0.

LINEA 400

Usando le routine di predisposizione di BORDER e degli attributi permanenti presentate nel Capitolo 1, si può convertire questa linea nella routine da 608A a 609D.
La chiamata a CLS pone i colori sullo schermo esattamente come nel BASIC.

LINEA 410

Questa linea predispone il valore iniziale del punteggio massimo. Supponendo che questo non possa mai essere maggiore di 65535, useremo per immagazzinarlo i primi due byte del buffer di stampa (5B00/01–23296/7). L'azzeramento viene agevolmente ottenuto tramite LD HL, 0000h e LD(5B00), HL, come si vede nella routine da 609C a 60A1.

LINEA 420

DATA per la visualizzazione = 60A4–60C8
Routine PRINT DATA = 60C9–60D6

LINEA 430

Predispone il valore iniziale delle "vite" disponibili, del "punteggio" e

delle "case". Le vite ancora disponibili possono essere immagazzinate sotto forma di un byte, il punteggio in due byte e le case in un byte. Questi valori vengono immagazzinati nel buffer di stampa. La cella 5B02 contiene "vite", 5B03/4 "punteggio" e 5B05 "case". I valori iniziali vengono immagazzinati dalla routine da 60D7 a 60E5.

LINEA 440

Questa linea riporta a zero la velocità ponendo istruzioni RET nei punti opportuni della subroutine di roll della linea. Routine da 60E6 a 60F0.

LINEA 450

DATA per la visualizzazione = 60F4–6128
Routine PRINT DATA = 6129–6136

LINEA 455

Si controlla che "case" sia diverso da 0, per sapere se occorre rivisualizzare lo schermo oppure lasciarlo così com'è vuotando le "case". Il valore contenuto in 5B05 viene testato tramite l'istruzione AND A. Se questa setta il flag zero, si salta alla routine di visualizzazione del terreno di gioco; in caso contrario, si passa direttamente alla routine di gioco vera e propria. Routine da 6137 a 6140.

LINEE 460–640

Questa è la routine più lunga del programma e dimostra la noiosità dell'operazione di visualizzazione del terreno di gioco.
DATA per la visualizzazione = 6141–63D1
Routine PRINT DATA = 63D2–63DF

LINEA 650

Noterete che questa routine di visualizzazione contiene la variabile "vite", quindi sarà preventivamente necessario porne il valore aggiornato fra i DATA da visualizzare. A ciò provvede la routine da 64E0 a 63E9 che legge il valore contenuto in 5B02, vi somma 30h in modo da convertirlo in codice ASCII e lo pone in 6401.
DATA per la visualizzazione = 63EA–640F
Routine PRINT DATA = 6410–641D

LINEA 660

Predispone i valori iniziali x1, y1 della posizione dell'omino sullo schermo e quelli della nuova posizione x2, y2, usati dalla routine SCREEN$. Viene utilizzato il buffer di stampa:

x1 = 5B06
y1 = 5B07
x2 = 5B08
y2 = 5B09

L'operazione è agevolmente eseguita da LD HL, 1014h: LD (5B06), HL: LD (5B08), HL. Noterete la comodità di usare la rappresentazione esadecimale anziché decimale: H contiene 10h (16d) e L contiene 14h (20d). Routine da 641E a 6426.

LINEA 670

Poiché vogliamo visualizzare un carattere in posizione x1, y1, è più conveniente usare LD A, n: RST 10h invece della routine PRINT DATA. Routine da 6427 a 6445.

LINEA 680

Routine da 6446 a 6448. Lasceremo la routine ROLL all'indirizzo 7E27 (32295).

LINEA 690

Useremo le routine descritte nel Capitolo 8 per trovare SCREEN$ (x2, y2) e ottenere i valori opportuni nei registri A,B,C,D,E. Il carattere viene poi prelevato tramite LD A, (DE) e confrontato con 20h (32d). Ricordate la necessità di ripulire l'area di lavoro prima di continuare l'esecuzione. Routine da 6449 a 645D. Se il carattere è diverso da uno spazio, si salta alla routine HIT che inizia in 6548h. Nel BASIC il controllo viene ceduto alla linea 880.

LINEA 880

Controlliamo che l'omino non si trovi sulla linea superiore dello schermo, che non sia, cioè, arrivato in una delle "case". In codice macchina otteniamo il valore x2 tramite LD A, (5B08) e verifichiamo se sia 0. In tal caso, il controllo viene ceduto alla routine che inizia in 66A6. Routine da 645E a 6465. Il loop principale del BASIC salta alla linea 1050.

LINEA 1050

Analoga alla linea 670. Il carattere viene visualizzato in x2, y2. Routine da 6466 a 6484.

LINEA 1060

Copia in 5B06/7 il valore contenuto in 5B08/9. Routine da 6485 a 648A.

LINEA 1070

In questo punto, per migliorare il gioco, non si è seguito fedelmente il BASIC. Invece di acquisire dati dalla tastiera tramite la variabile LAST KEY, sfruttiamo l'istruzione IN A,(C), che permette la lettura contemporanea di tutti i tasti e quindi anche i movimenti in diagonale. Per leggere il tasto 1, dovremo operare sulla semiriga 1 – 5 sfruttando la routine descritta nel Capitolo 5. Routine da 648B a 6493.

LINEA 1080

Non è necessaria una particolare precisione nella conversione di durata e tono del BEEP. Invece di CALCULATOR, useremo la routine BEEP in 03B5 (949d). Routine da 6494 a 649C.

LINEA 1090

Aggiorniamo la posizione x2 dell'omino decrementando di due il valore contenuto in 5B08. Si incrementa anche il punteggio sommando 5 al valore contenuto in 5B03/4. Routine da 649D a 64AF. Per visualizzare il punteggio useremo le routine STACK BC e PRINT VALUE ON STACK, avendo cura di definire preventivamente i parametri di PRINT AT. Routine da 64B1 a 64C8.

LINEA 1100

Sfruttando l'istruzione IN A, (C) leggiamo la tastiera e aggiorniamo il valore di y2 (5B09). Routine da 64C9 a 64FI. Ho aggiunto una routine che verifica l'eventuale pressione del tasto 6 per permettere il ritorno al BASIC (necessario per registrare il programma finalmente funzionante). Routine da 64F2 a 64F4. Abbiamo così completato il loop principale; noterete però che il programma è tanto veloce da rendere praticamente invisibile il nostro omino. Abbiamo quindi bisogno di un loop di ritardo. Invece di sprecare il tempo, ho aggiunto un effetto sonoro che simula il rumore del traffico. Routine da 64F5 a 650B. JP 6427 ci riporterà all'inizio del loop principale (linea 670).

ROUTINE HIT

LINEA 700

Introduciamo la variabile a = x2, che verrà immagazzinata nella locazione 5CB0 (un indirizzo non sfruttato fra le variabili di sistema). La variabile b sfruttata come contatore del loop FOR/NEXT può essere sostituita con LD B, 19h (25d). Questo valore viene posto in 5CB1 per essere sfruttato dalla successiva routine. Routine da 6548 a 6561; l'istruzione GOSUB 25 è sostituita da CALL 650F.

LINEA 25

La linea è facilmente convertibile usando CALCULATOR per porre 0,1 sullo stack di calcolo. Le istruzioni LD HL, (5CB0): LD A,H: SUBL eseguono b−a. I valori di a e di b sono immagazzinati consecutivamente in 5CB0 e 5CB1. Sfruttiamo poi la routine STACK A in 2D28 e la routine BEEP in 03F8. Routine da 650F a 6521.

LINEA 30

Sfruttiamo ancora le istruzioni LD A, n: RST 10h, questa volta con due variabili nella routine PRINT. Routine da 6522 a 6547.

LINEA 730

La variabile a è già immagazzinata in 5CB0, possiamo quindi eseguire un loop che incrementi di due questa variabile ad ogni passo, fino al raggiungimento del valore 14h (20d). Routine da 6562 a 6573.

LINEA 740

In numero di "vite" rimanenti (memorizzato in 5B02) viene diminuito di uno. Per ottenerne la visualizzazione, predisponiamo i parametri di PRINT AT, quindi sommiamo 30h (48d) "vite" (per ottenerne la conversione in ASCII) e sfruttiamo RST 10h. Routine da 6574 a 658E.

LINEA 750

Routine da 658F a 6593. Resetta x2 (5B08) a 14h (20d).

LINEA 760

Routine da 6594 a 659A. È estremamente semplice. Se "vite" è diverso da

zero (se cioè non abbiamo consumato tutte le "vite" a nostra disposizione) si salta in 6446, all'inizio del loop principale.

ROUTINE DI FINE GIOCO

LINEA 770

Si confronta il punteggio ottenuto con quello massimo. Vengono sfruttate le istruzioni SBC HL, DE e si controlla lo stato del flag di carry. Routine da 659B a 65A6. Se non si è riusciti a superare il punteggio massimo si salta a 6611.

LINEA 780

Il punteggio ottenuto (memorizzato in 5B03/4) viene trasferito nelle locazioni 5B00/1, destinate a contenere il punteggio massimo. Routine da 65A7 a 65AC. Per visualizzare il nuovo record predisponiamo i parametri di PRINT AT e sfruttiamo le routine STACK VALUE IN BC e PRINT TOP VALUE ON CALCULATOR STACK. Routine da 65AD a 66C2.

LINEE 790–800

DATA per la visualizzazione = 65C5–6610
Routine PRINT DATA = 6611–6619

LINEE 840–850

Attendiamo (sfruttando la variabile LAST KEY) che venga premuto un tasto; se questo non è né "s" né "n", ritorniamo nel loop di attesa. Routine da 661A a 6632.

LINEA 860

Sfruttiamo le istruzioni LD A,n:RST 10h; tramite JP 60C9 ritorniamo al loop principale rivisualizzando il terreno di gioco con il punteggio massimo invariato. Routine da 6633 a 6655.

ROUTINE HOME

LINEA 890

Posti fra i DATA i valori di x1, y1, x2, y2, sfruttiamo la routine PRINT DATA.

DATA per la visualizzazione = 669A–66A5
Predisposizione valori di x1, y1, x2, y2 = 66A6–66B1
Routine PRINT DATA = 66B2–66BF

LINEA 900

Lavorando in codice macchina non è necessario ripristinare (RESTORE) il puntatore all'inizio dei DATA. La conversione non è pertanto necessaria.

LINEE 910–920

BEEP FANFARA: call 666A
La routine è ampiamente descritta nel Capitolo 7 e non richiede ulteriori commenti. Routine da 66C0 a 66C2.

LINEA 930

I valori di "case" e "punteggio" vengono prima aggiornati e poi visualizzati tramite le routine STACK VALUE IN BC e PRINT TOP VALUE ON STACK. Routine di aggiornamento da 66C3 a 66D3. Routine di visualizzazione punteggio da 66D4 a 66E9.

LINEA 950

Si controlla l'eventuale riempimento delle quattro "case". L'operazione si esegue più facilmente in codice macchina che in BASIC. Routine da 66EA a 66F1. Il contatore "case" viene posto nell'accumulatore tramite LD A, (5B05) e mascherato mediante AND 03h. Se il risultato è non-zero, l'originario valore di "case" non era multiplo di quattro; in tal caso, torniamo nel loop principale: JP 641E.

LINEE 960–980

Queste routine sono usate per aumentare la velocità del gioco quando "case" = 4, 8, 12. Ciò è ottenuto rimuovendo le istruzioni di RETURN nella subroutine di roll, permettendo così ad alcune linee di essere mosse di un pixel in più; linea 960 = 66F2–66FF; linea 970 = 6700–6709; linea 980 = 670A–6712.

LINEA 985

Controlliamo che "case" non sia maggiore di 36; in tal caso non viene eseguita la routine di aggiunta di un ragno e si ritorna al loop principale.

L'operazione è semplice: si sottrae 37 dal valore di "case" e se il flag di carry = 0, si salta in 6129. Routine da 6714 a 6718.

LINEA 990

La routine è stata descritta nel Capitolo 4 e non richiede ulteriori spiegazioni. In questo caso usiamo il valore modificato di SEED in BC e mascheriamo con 1Fh (31d): LD A,C: AND 1Fh, quindi sottraiamo due e controlliamo il flag di carry. Questo metodo è stato scelto perché non è necessario un numero veramente casuale. Routine da 6719 a 6740.

LINEA 1000

Routine 6741. È necessario solo incrementare il valore in a.

LINEA 1005

Si controlla che a non sia maggiore di 31, cioè al di fuori dello schermo; in tal caso esso viene riportato a 0. Routine da 6742 a 6746.

LINEA 1010

Si controlla, tramite la routine SCREEN$, che la posizione di schermo (10,a) sia libera. Occorre porre il valore a nel registro B e 0ah (10d) nel registro C. Vengono quindi chiamate le routine STACK VALUE IN BC e SCREEN$; i parametri sono poi rimossi dallo stack e posti nei registri A, B, C, D, E. Si controlla infine se SCREEN$ (10,a) = 20h (32d). In tal caso riprendiamo il valore originale di a (POP AF) e continuiamo. Se la routine SCREEN$ indica che la posizione non è libera, eseguiamo JR 6741, passando così alla successiva posizione di schermo. Routine da 6747 a 675E.

LINEA 1020

Questa è la ripetizione della routine precedente, tranne che il valore di a viene incrementato prima dell'esecuzione della routine SCREEN$. Routine da 675F a 6776. Da notare che il valore di a viene salvato (6760 PUSH AF) prima della routine SCREEN$ per essere usato nella routine successiva.

LINEA 1030

Il valore a viene ripreso (POP AF) e diminuito. Viene poi posto fra i dati da visualizzare e viene chiamata la routine PRINT. Routine da 6777 a 6792.

LINEA 1030

È una ripetizione della routine BEEP FANFARA e richiede solo una CALL 666A. Routine da 6793 a 6795.

LINEA 1040

Torniamo infine nel loop principale usando JP 6129. Routine da 6796 a 6798.

Come potete vedere, è estremamente semplice convertire un programma BASIC in codice macchina. Dovete solo procedere un passo alla volta e progettare il programma tenendo presente il codice macchina. Senz'altro avrete notato che la notazione esadecimale è molto più conveniente di quella decimale, specialmente nelle routine che richiedono di porre in BC i parametri di linea/colonna. Considerate la routine di roll a destra/sinistra. Gli indirizzi di schermo espressi in decimale non significano apparentemente molto ma, se ne considerate gli equivalenti esadecimali, vedrete un tipo di trama in grado di fornire maggiori indicazioni su quanto sta effettivamente avvenendo.
A questo punto non vi resta che iniziare a immaginare vostri originali e divertenti programmi e iniziarne la conversione in codice macchina.
Vi auguro buona fortuna e spero che possiate trascorrere felicemente molte ore a programmare senza combinare troppi disastri.

Programma CROSS in versione codice macchina

```
org 43000 24001

5DC1h
defb $0f $12 $22 $7f $ff $ff $28
defb $10
defb $80 $40 $20 $fe $fe $ff $28
defb $10
defb $7f $7f $7f $7f $7f $ff $15
defb $08
defb $ff $fe $fe $fe $ff $ff $40
defb $80
defb $00 $f8 $c4 $c4 $fe $fe $28
defb $10
defb $18 $18 $24 $7e $3c $5a $a5
defb $42
```

```
defb $38 $28 $92 $72 $38 $38 $28
defb $6c

defb $01 $02 $04 $7f $7f $ff $14
defb $08
defb $f0 $48 $44 $fe $ff $ff $14
defb $08
defb $00 $1f $23 $23 $7f $7f $14
defb $08
defb $7f $7f $7f $7f $ff $ff $02
defb $01
defb $fe $fe $fe $fe $ff $a8
defb $10
defb $10 $29 $c7 $00 $26 $00 $00
defb $00
defb $00 $44 $ff $44 $44 $ff $44
defb $00
defb $00 $22 $55 $8f $97 $a3 $a0
defb $00
defb $00 $44 $aa $f1 $e9 $c5 $05
defb $00
defb $10 $10 $10 $fe $3f $1f $0f
defb $07
defb $00 $00 $00 $00 $1e $ff $ff

defb $ff
defb $60 $7c $54 $78 $7f $ff $fe
defb $7c
defb $00 $00 $03 $02 $0f $3f $ff
defb $00
defb $06 $0c $98 $f0 $e0 $55 $ff
defb $00



5E69h
24169 ED 5B 7B 5C ld de,($5c7b)
24173 21 C1 5D    ld hl,$5dc1
24176 01 A8 00    ld bc,$00a8
24179 ED B0       ldir
24181 18 25       jr +$25



5E77h
defb $16 $0b $05 $56 $75 $6f $69

defb $20
defb $6c $65 $20 $69 $73 $74 $72
defb $75
```

```
defb $7a $69 $6f $6e $69 $20 $3f
defb $16
defb $0d $0d $28 $73 $29 $69 $16
defb $0f
defb $0d $28 $6e $29 $6f


5E9Ch
24220 3E 02       ld a,$02
24222 CD 01 16    call $1601
24225 11 77 5E    ld de,$5e77
24228 01 25 00    ld bc,$0025
24231 CD 3C 20    call $203c
24234 76          halt
24235 FD CB 01 6E bit 5,(iy+$01
24239 28 F9       jr z,-$07
24241 3A 08 5C    ld a,($5c08)

24244 FE 73       cp $73
24246 C2 8A 60    jp nz,$608a


5EB9h
24249 3E 02       ld a,$02
24251 CD 01 16    call $1601
24254 CD 6B 0D    call $0d6b
24257 C3 75 5F    jp $5f75


5EC4
defb $16 $00 $0b $4f $47 $47 $45
defb $54
defb $54 $4f $0d $0d $47 $75 $69
defb $64
defb $61 $72 $65 $20 $75 $6e $20
defb $96
defb $20 $61 $74 $74 $72 $61 $76

defb $65
defb $72 $73 $6f $20 $73 $74 $72
defb $61
defb $64 $61 $20 $65 $66 $69 $75
defb $6d
defb $65 $2c $20 $65 $76 $69 $74
defb $61
defb $6e $64 $6f $20 $90 $91 $20
defb $92
```

```
defb $93 $94 $20 $95 $20 $a0 $a1
defb $a1
defb $a2 $20 $a3 $a4 $0d $0d $75
defb $6e
defb $20 $9e $9f $20 $70 $61 $74
defb $74
defb $75 $67 $6c $69 $61 $20 $6c
defb $27
defb $69 $73 $6f $6c $61 $20 $63
defb $65
defb $6e $74 $72 $61 $6c $65 $20
defb $0d

defb $0d $44 $6f $76 $65 $74 $65
defb $20
defb $72 $69 $65 $6d $70 $69 $72
defb $65
defb $20 $6c $65 $20 $34 $20 $43
defb $41
defb $53 $45 $2c $63 $69 $6f $65
defb $27
defb $69 $76 $75 $6f $74 $69 $20
defb $73
defb $75 $6c $20 $62 $6f $72 $64
defb $6f
defb $20 $73 $75 $70 $65 $72 $69
defb $6f
defb $72 $65 $20 $9d $9d $20 $9d
defb $9d $0d



5F75h
24437 3E 02       ld a,$02

24439 CD 01 16    call $1601
24442 11 C4 5E    ld de,$5ec4
24445 01 B1 00    ld bc,$00b1
24448 CD 3C 20    call $203c
24451 C3 F8 5F    jp $5ff8



5F86h
defb $0d $55 $6e $61 $20 $76 $6f
defb $6c
defb $74 $61 $20 $72 $69 $65 $6d
defb $70
defb $69 $74 $65 $2c $20 $6c $65
defb $20
```

```
defb $34 $20 $43 $41 $53 $45 $20
defb $73
defb $69 $73 $76 $75 $6f $74 $65
defb $72
defb $61 $6e $6e $6f $2c $6c $61
defb $20

defb $76 $65 $6c $6f $63 $69 $74
defb $61
defb $27 $20 $73 $61 $6c $69 $72
defb $61
defb $27 $65 $20 $76 $65 $72 $72
defb $61
defb $27 $20 $61 $67 $67 $69 $75
defb $6e
defb $74 $6f $20 $75 $6e $20 $61
defb $6c
defb $74 $72 $6f $20 $9e $9f $2e
defb $16
defb $12 $09 $50 $72 $65 $6d $69
defb $20
defb $75 $6e $20 $74 $61 $73 $74
defb $6f
defb $20 $20


5FF8h

24568 11 86 5F     ld de,$5f86
24571 01 71 00     ld bc,$0071
24574 CD 3C 20     call $203c
24577 FD CB 01 AE  res 5,(iy+$01
24581 76           halt
24582 FD CB 01 6E  bit 5,(iy+$01
24586 28 F9        jr z,-$07


600Ch
24588 06 18        ld b,$18
24590 CD 44 0E     call $0e44
24593 C3 76 60     jp $6076


6014h
defb $16 $07 $0b $43 $4f $4d $41
defb $4e
defb $44 $49 $3a $16 $09 $04 $20
```

```
defb $20
defb $5e $20 $20 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $20
defb $20
defb $20 $20 $20 $3c $30 $3e $12
defb $01
defb $16 $0b $06 $31 $12 $00 $20
defb $32
defb $20 $33 $20 $34 $20 $35 $20
defb $36
defb $20 $37 $20 $38 $30 $12 $01
defb $39
defb $12 $00 $20 $12 $01 $30 $12
defb $00
defb $16 $12 $03 $50 $65 $72 $20
defb $67
defb $69 $6f $63 $61 $72 $65 $20
defb $70
defb $72 $65 $6d $69 $20 $75 $6e
defb $20 $74 $61 $73 $74 $6f
```

```
6076h
24694 11 14 60     ld de,$6014
24697 01 62 00     ld bc,$0062
24700 CD 3C 20     call $203c
24703 FD CB 01 AE  res 5,(iy+$01
24707 76           halt
24708 FD CB 01 6E  bit 5,(iy+$01
24712 28 F9        jr z,-$07
```

```
608Ah
24714 3E 05        ld a,$05
24716 CD 9B 22     call $229b
24719 3E 68        ld a,$68
24721 32 8D 5C     ld ($5c8d),a
24724 3E 02        ld a,$02
24726 CD 01 16     call $1601

24729 CD 6B 0D     call $0d6b
```

```
609Ch
24732 21 00 00     ld hl,$0000
```

```
24735 22 00 5B     ld (#5b00),hl
24738 18 25        jr +#25


60A4h
defb #11 #04 #16 #0a #00 #20 #20
defb #20
defb #20 #20 #20 #20 #20 #20 #20
defb #20
defb #20 #20 #20 #20 #9e #9f #20
defb #20
defb #20 #20 #20 #20 #20 #20 #20
defb #20
defb #20 #20 #20 #20 #20


60C9h
24777 3E 02        ld a,#02
24779 CD 01 16     call #1601
24782 11 A4 60     ld de,#60a4
24785 01 25 00     ld bc,#0025
24788 CD 3C 20     call #203c


60D7h
24791 3E 09        ld a,#09
24793 32 02 5B     ld (#5b02),a
24796 21 00 00     ld hl,#0000
24799 22 03 5B     ld (#5b03),hl
24802 AF           xor a
24803 32 05 5B     ld (#5b05),a


60E6h
24806 3E C9        ld a,#c9
24808 32 A9 7E     ld (#7ea9),a
24811 32 C2 7E     ld (#7ec2),a
24814 32 D5 7E     ld (#7ed5),a
24817 C3 29 61     jp #6129


60F4h
defb #16 #00 #00 #11 #04 #9d #9d
defb #9d
defb #9d #11 #07 #20 #11 #04 #9d
```

```
defb $9d
defb $9d $9d $9d $11 $07 $20
defb $11
defb $04 $9d $9d $9d $9d $9d $9d
defb $9d
defb $11 $07 $20 $11 $04 $9d $9d
defb $9d

defb $9d $9d $9d $9d $11 $07 $20
defb $11
defb $04 $9d $9d $9d $9d
```

```
6129h
24873 3E 02       ld a,$02
24875 CD 01 16    call $1601
24878 11 F4 60    ld de,$60f4
24881 01 35 00    ld bc,$0035
24884 CD 3C 20    call $203c
```

```
6137h
24887 3A 05 5B    ld a,($5b05)
24890 A7          and a
24891 C2 1E 64    jp nz,$641e
24894 C3 D2 63    jp $63d2
```

```
6141h
defb $11 $04 $10 $05 $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $8c $8c $8c
defb $8c
defb $8c $8c $8c $8c $11 $05 $10
defb $00
defb $20 $a0 $a1 $a1 $a2 $20 $20
defb $20
defb $a0 $a1 $a1 $a1 $a2 $20 $20
defb $20
defb $20 $a0 $a1 $a1 $a2 $20 $20
defb $20
defb $a0 $a1 $a1 $a2 $20 $20 $20
defb $20
```

```
defb $10 $07 $20 $20 $20 $9c $9c
defb $9c
defb $20 $20 $20 $20 $20 $9c $9c
defb $9c
defb $9c $9c $20 $20 $20 $20 $20
defb $9c
defb $9c $9c $20 $20 $20 $20 $20
defb $20
defb $9c $10 $02 $20 $20 $a3 $a4
defb $20
defb $20 $20 $20 $20 $a3 $a4 $20
defb $20
defb $20 $20 $20 $a3 $a4 $20 $20
defb $20
defb $20 $a3 $a4 $20 $20 $20 $20
defb $20
defb $20 $a3 $a4 $20 $10 $07 $9c
defb $9c
defb $20 $20 $20 $9c $20 $20 $20
defb $20
defb $20 $20 $9c $9c $9c $9c $9c

defb $20
defb $20 $20 $20 $20 $20 $20 $20
defb $9c
defb $9c $9c $20 $20 $20 $20 $10
defb $01
defb $a1 $a2 $20 $20 $20 $20 $20
defb $a0
defb $a1 $a1 $a2 $20 $20 $20 $20
defb $20
defb $20 $a0 $a1 $a1 $a1 $a1 $a2
defb $20
defb $20 $20 $20 $20 $a0 $a1 $a1
defb $a1
defb $10 $07 $20 $20 $20 $9c $9c
defb $20
defb $20 $9c $9c $9c $9c $9c $20
defb $20
defb $20 $9c $9c $20 $20 $20 $20
defb $20
defb $20 $9c $9c $9c $20 $20 $20
defb $20

defb $20 $20 $10 $00 $a4 $20 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $a3
defb $a4
defb $20 $20 $20 $20 $20 $a3 $a4
defb $20
```

```
defb $20 $20 $20 $20 $20 $a3 $a4
defb $20
defb $20 $20 $20 $a3 $11 $04 $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $11
defb $00
defb $10 $07 $16 $0b $00 $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d

defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $10 $03
defb $20
defb $90 $91 $20 $95 $20 $20 $90
defb $91
defb $20 $20 $92 $93 $92 $93 $94
defb $20
defb $95 $20 $95 $20 $20 $20 $92
defb $93
defb $94 $20 $20 $20 $20 $20 $20
defb $10
defb $07 $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d

defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $10 $05 $20 $90 $91 $20
defb $20
defb $20 $20 $90 $91 $20 $20 $20
defb $20
defb $90 $91 $20 $20 $90 $91 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $90
defb $91
defb $20 $20 $20 $10 $07 $3d $3d
defb $3d
```

```
defb $3d $3d $3d $3d $3d $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $3d $3d
defb $3d
defb $3d $3d $3d $3d $3d $10 $04
defb $98
defb $20 $20 $20 $20 $20 $97 $98

defb $20
defb $20 $20 $97 $98 $20 $20 $20
defb $20
defb $20 $20 $20 $97 $98 $20 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $97
defb $10
defb $07 $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $2d $2d $2d $2d $2d $2d
defb $2d
defb $2d $10 $06 $9a $9b $20 $20
defb $20
defb $97 $98 $20 $20 $97 $98 $20
defb $95
defb $20 $20 $95 $20 $20 $99 $9a
defb $9b

defb $9a $9b $20 $20 $20 $95 $20
defb $20
defb $20 $20 $99 $11 $04 $10 $00
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $9d
defb $9d $9d $9d $9d $9d $9d $9d
defb $11
defb $04 $20 $20 $20 $20 $20 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $20
defb $20
```

```
defb $20 $20 $20 $20 $20 $20 $20
defb $20 $20



63D2h
25554 3E 02        ld a,$02
25556 CD 01 16     call $1601
25559 11 41 61     ld de,$6141
25562 01 91 02     ld bc,$0291
25565 CD 3C 20     call $203c



63E0h
25568 3A 02 5B     ld a,($5b02)
25571 C6 30        add a,$30
25573 32 01 64     ld ($6401),a
25576 18 26        jr +$26



63EAh
defb $11 $01 $10 $07 $20 $53 $43

defb $4f
defb $52 $45 $20 $16 $15 $0b $20
defb $4d
defb $45 $4e $20 $11 $05 $10 $00
defb $39
defb $11 $01 $10 $07 $20 $48 $49
defb $2d
defb $53 $43 $4f $52 $45 $20



6410h
25616 3E 02        ld a,$02
25618 CD 01 16     call $1601
25621 11 EA 63     ld de,$63ea
25624 01 26 00     ld bc,$0026
25627 CD 3C 20     call $203c



641Fh

25630 21 14 10     ld hl,$1014
25633 22 06 5B     ld ($5b06),hl
25636 22 08 5B     ld ($5b08),hl
```

```
6427h
25639 3E 02       ld a,$02
25641 CD 01 16    call $1601
25644 3E 11       ld a,$11
25646 D7          rst $10
25647 3E 08       ld a,$08
25649 D7          rst $10
25650 3E 10       ld a,$10
25652 D7          rst $10
25653 3E 08       ld a,$08
25655 D7          rst $10
25656 3E 16       ld a,$16
25658 D7          rst $10
25659 3A 06 5B    ld a,($5b06)
25662 D7          rst $10

25663 3A 07 5B    ld a,($5b07)
25666 D7          rst $10
25667 3E 20       ld a,$20
25669 D7          rst $10



6446h
25670 CD 27 7E    call $7e27



6449h
25673 ED 4B 08 5B ld bc,($5b08)
25677 CD 38 25    call $2538
25680 CD F1 2B    call $2bf1
25683 1A          ld a,(de)
25684 F5          push af
25685 CD BF 16    call $16bf
25688 F1          pop af
25689 FE 20       cp $20

25691 C2 48 65    jp nz,$6548



645Eh
25694 3A 08 5B    ld a,($5b08)
25697 FE 00       cp $00
25699 CA A6 66    jp z,$66a6



6466h
25702 3E 02       ld a,$02
```

```
25704 CD 01 16     call $1601
25707 3E 11        ld a,$11
25709 D7           rst $10
25710 3E 08        ld a,$08
25712 D7           rst $10
25713 3E 10        ld a,$10
25715 D7           rst $10
25716 3E 08        ld a,$08

25718 D7           rst $10
25719 3E 16        ld a,$16
25721 D7           rst $10
25722 3A 08 5B     ld a,($5b08)
25725 D7           rst $10
25726 3A 09 5B     ld a,($5b09)
25729 D7           rst $10
25730 3E 96        ld a,$96
25732 D7           rst $10


6485h
25733 2A 08 5B     ld hl,($5b08)
25736 22 06 5B     ld ($5b06),hl


648Bh
25739 01 FE F7     ld bc,$f7fe
25742 ED 78        in a,(c)

25744 CB 47        bit 0,a
25746 20 1C        jr nz,+$1c


6494h
25748 21 32 00     ld hl,$0032
25751 11 05 00     ld de,$0005
25754 CD B5 03     call $03b5


649Dh
25757 3A 08 5B     ld a,($5b08)
25760 3D           dec a
25761 3D           dec a
25762 32 08 5B     ld ($5b08),a
25765 2A 03 5B     ld hl,($5b03)
25768 23           inc hl
```

```
25769 23           inc hl
25770 23           inc hl

25771 23           inc hl
25772 23           inc hl
25773 22 03 5B     ld ($5b03),hl
25776 00           nop
25777 3E 02        ld a,$02
25779 CD 01 16     call $1601
25782 3E 16        ld a,$16
25784 D7           rst $10
25785 3E 15        ld a,$15
25787 D7           rst $10
25788 3E 07        ld a,$07
25790 D7           rst $10
25791 ED 4B 03 5B  ld bc,($5b03)
25795 CD 2B 2D     call $2d2b
25798 CD E3 2D     call $2de3


64C9h
25801 01 FE EF     ld bc,$effe
25804 ED 78        in a,(c)

25806 CB 47        bit 0,a
25808 20 0E        jr nz,+$0e
25810 3A 09 5B     ld a,($5b09)
25813 FE 1F        cp $1f
25815 28 01        jr z,+$01
25817 3C           inc a
25818 32 09 5B     ld ($5b09),a
25821 C3 F5 64     jp $64f5
25824 CB 4F        bit 1,a
25826 20 0E        jr nz,+$0e
25828 3A 09 5B     ld a,($5b09)
25831 FE 00        cp $00
25833 28 01        jr z,+$01
25835 3D           dec a
25836 32 09 5B     ld ($5b09),a
25839 C3 F5 64     jp $64f5


64F2h
25842 CB 67        bit 4,a

25844 C8           ret z
```

```
64F5h
25845 06 0A       ld b,$0a
25847 11 78 00    ld de,$0078
25850 21 01 00    ld hl,$0001
25853 E5          push hl
25854 D5          push de
25855 C5          push bc
25856 CD B5 03    call $03b5
25859 C1          pop bc
25860 D1          pop de
25861 E1          pop hl
25862 7D          ld a,l
25863 3C          inc a
25864 6F          ld l,a
25865 00          nop
25866 10 F1       djnz -$0f
25868 C3 27 64    jp $6427



650Fh
25871 EF          rst $28


6510h
defb $34 $ea $23 $d7 $0a $3d $38


6517h
25879 2A B0 5C    ld hl,($5cb0)
25882 7C          ld a,h
25883 95          sub l
25884 CD 28 2D    call $2d28
25887 CD F8 03    call $03f8
25890 3E 02       ld a,$02
25892 CD 01 16    call $1601

25895 3E 15       ld a,$15
25897 D7          rst $10
25898 3E 01       ld a,$01
25900 D7          rst $10
25901 3E 11       ld a,$11
25903 D7          rst $10
25904 3E 08       ld a,$08
25906 D7          rst $10
25907 3E 10       ld a,$10
25909 D7          rst $10
25910 3E 08       ld a,$08
```

```
25912 D7          rst $10
25913 3E 16       ld a,$16
25915 D7          rst $10
25916 3A B0 5C    ld a,($5cb0)
25919 D7          rst $10
25920 3A 09 5B    ld a,($5b09)
25923 D7          rst $10
25924 3E 96       ld a,$96
25926 D7          rst $10
25927 C9          ret



6548h
25928 3A 08 5B    ld a,($5b08)
25931 32 B0 5C    ld ($5cb0),a
25934 06 19       ld b,$19
25936 78          ld a,b
25937 C5          push bc
25938 32 B1 5C    ld ($5cb1),a
25941 CD 0F 65    call $650f
25944 CD 0F 65    call $650f
25947 C1          pop bc
25948 04          inc b
25949 78          ld a,b
25950 FE 23       cp $23
25952 20 EF       jr nz,-$11


6562h

25954 CD 0F 65    call $650f
25957 CD 0F 65    call $650f
25960 3A B0 5C    ld a,($5cb0)
25963 3C          inc a
25964 3C          inc a
25965 32 B0 5C    ld ($5cb0),a
25968 FE 14       cp $14
25970 20 EE       jr nz,-$12


6574h
25972 3A 02 5B    ld a,($5b02)
25975 3D          dec a
25976 32 02 5B    ld ($5b02),a
25979 3E 02       ld a,$02
25981 CD 01 16    call $1601
```

```
25984 3E 16        ld a,$16
25986 D7           rst $10
25987 3E 15        ld a,$15
25989 D7           rst $10

25990 3E 10        ld a,$10
25992 D7           rst $10
25993 3A 02 5B     ld a,($5b02)
25996 C6 30        add a,$30
25998 D7           rst $10


658Fh
25999 3E 14        ld a,$14
26001 32 08 5B     ld ($5b08),a


6594h
26004 3A 02 5B     ld a,($5b02)
26007 A7           and a
26008 C2 46 64     jp nz,$6446


659Bh
26011 AF           xor a
26012 2A 03 5B     ld hl,($5b03)
26015 ED 5B 00 5B  ld de,($5b00)
26019 ED 52        sbc hl,de
26021 38 6A        jr c,+$6a


65A7h
26023 2A 03 5B     ld hl,($5b03)
26026 22 00 5B     ld ($5b00),hl
26029 E5           push hl
26030 C1           pop bc
26031 CD 2B 2D     call $2d2b
26034 3E 02        ld a,$02
26036 CD 01 16     call $1601
26039 3E 16        ld a,$16
26041 D7           rst $10
26042 3E 15        ld a,$15
26044 D7           rst $10

26045 3E 1B        ld a,$1b
```

```
26047 D7          rst $10
26048 CD E3 2D    call $2de3
26051 18 4C       jr +$4c


65C5h
defb $12 $01 $11 $07 $16 $0c $00
defb $20
defb $20 $20 $20 $20 $20 $20 $20
defb $20
defb $20 $20 $47 $41 $4d $45 $20
defb $20
defb $4f $56 $45 $52 $20 $20 $20
defb $20
defb $20 $20 $20 $20 $20 $20 $20
defb $16
defb $0e $00 $12 $00 $56 $75 $6f
defb $69
defb $20 $67 $69 $6f $63 $61 $72

defb $65
defb $20 $61 $6e $63 $6f $72 $61
defb $20
defb $3f $20 $28 $73 $29 $69 $20
defb $20
defb $28 $6e $29 $6f


6611h
26129 11 C5 65    ld de,$65c5
26132 01 4C 00    ld bc,$004c
26135 CD 3C 20    call $203c


661Ah
26138 76          halt
26139 FD CB 01 6E bit 5,(iy+$01
26143 28 F9       jr z,-$07
26145 FD CB 01 AE res 5,(iy+$01

26149 3A 08 5C    ld a,($5c08)
26152 FE 6E       cp $6e
26154 20 03       jr nz,+$03
26156 CD 00 00    call $0000
26159 FE 73       cp $73
26161 20 E7       jr nz,-$19
```

```
6633h
26163 3E 02       ld a,$02
26165 CD 01 16    call $1601
26168 3E 11       ld a,$11
26170 D7          rst $10
26171 3E 05       ld a,$05
26173 D7          rst $10
26174 3E 16       ld a,$16
26176 D7          rst $10
26177 3E 15       ld a,$15
26179 D7          rst $10
26180 3E 07       ld a,$07

26182 D7          rst $10
26183 3E 20       ld a,$20
26185 D7          rst $10
26186 3E 20       ld a,$20
26188 D7          rst $10
26189 3E 20       ld a,$20
26191 D7          rst $10
26192 3E 20       ld a,$20
26194 D7          rst $10
26195 C3 C9 60    jp $60c9


6656h
defb $00 $00 $00 $00 $00 $00 $ed
defb $0b
defb $ed $0b $f0 $10 $ec $0b $ec
defb $10
defb $ec $0b $ec $10



666Ah
26218 06 07       ld b,$07
26220 21 5C 66    ld hl,$665c
26223 7E          ld a,(hl)
26224 32 78 66    ld ($6678),a
26227 23          inc hl
26228 C5          push bc
26229 E5          push hl
26230 EF          rst $28
defb $34 $ec $4c $cc $cc $cc $38
26238 E1          pop hl
26239 7E          ld a,(hl)
26240 E5          push hl
26241 CD 28 2D    call $2d28
```

```
26244 CD F8 03     call $03f8
26247 E1           pop hl
26248 C1           pop bc
26249 23           inc hl
26250 10 E3        djnz -$1d
26252 3E 01        ld a,$01

26254 CD 28 2D     call $2d28
26257 3E 14        ld a,$14
26259 CD 28 2D     call $2d28
26262 CD F8 03     call $03f8
26265 C9           ret


669Ah
defb $11 $08 $10 $08 $16 $02 $04
defb $20
defb $16 $00 $04 $96


66A6h
26278 2A 06 5B     ld hl,($5b06)
26281 22 9F 66     ld ($669f),hl
26284 2A 08 5B     ld hl,($5b08)
26287 22 A3 66     ld ($66a3),hl
26290 3E 02        ld a,$02

26292 CD 01 16     call $1601
26295 11 9A 66     ld de,$669a
26298 01 0C 00     ld bc,$000c
26301 CD 3C 20     call $203c
26304 CD 6A 66     call $666a

          .

66C3h
26307 3A 05 5B     ld a,($5b05)
26310 3C           inc a
26311 32 05 5B     ld ($5b05),a
26314 2A 03 5B     ld hl,($5b03)
26317 11 32 00     ld de,$0032
26320 19           add hl,de
26321 22 03 5B     ld ($5b03),hl
26324 E5           push hl
26325 3E 02        ld a,$02
26327 CD 01 16     call $1601
26330 3E 16        ld a,$16
26332 D7           rst $10
```

```
26333 3E 15       ld a,$15
26335 D7          rst $10
26336 3E 07       ld a,$07
26338 D7          rst $10
26339 C1          pop bc
26340 CD 2B 2D    call $2d2b
26343 CD E3 2D    call $2de3


66EAh
26346 3A 05 5B    ld a,($5b05)
26349 E6 03       and $03
26351 C2 1E 64    jp nz,$641e


66F2h
26354 AF          xor a
26355 3A 05 5B    ld a,($5b05)
26358 FE 04       cp $04

26360 20 06       jr nz,+$06
26362 AF          xor a
26363 32 A9 7E    ld ($7ea9),a
26366 18 19       jr +$19


6700h
26368 FE 08       cp $08
26370 20 06       jr nz,+$06
26372 AF          xor a
26373 32 C2 7E    ld ($7ec2),a
26376 18 0F       jr +$0f


670Ah
26378 FE 0C       cp $0c
26380 20 06       jr nz,+$06
26382 AF          xor a
26383 32 D5 7E    ld ($7ed5),a

26386 18 05       jr +$05


6714h
26388 D6 25       sub $25
26390 D2 29 61    jp nc,$6129
```

```
5718h
26393 ED 4B 76 5C ld bc,($5c76)
26397 CD 2B 2D    call $2d2b
26400 EF          rst $28


6721h
defb $a1 $0f $34 $37 $16 $04 $34
defb $80
defb $41 $00 $00 $80 $32 $02 $a1

defb $03 $31 $38


6733h
26419 CD A2 2D    call $2da2
26422 ED 43 76 5C ld ($5c76),bc
26426 79          ld a,c
26427 E6 1F       and $1f
26429 D6 02       sub $02
26431 38 05       jr c,+$05


6741h
26433 3C          inc a


6742h
26434 FE 20       cp $20

26436 20 01       jr nz,+$01
26438 AF          xor a


6747h
26439 F5          push af
26440 47          ld b,a
26441 0E 0A       ld c,$0a
26443 CD 38 25    call $2538
26446 CD F1 2B    call $2bf1
26449 1A          ld a,(de)
26450 F5          push af
26451 CD BF 16    call $16bf
26454 F1          pop af
26455 FE 20       cp $20
26457 28 03       jr z,+$03
```

```
26459 F1          pop af
26460 18 E3       jr -$1d
26462 F1          pop af



675Fh
26463 3C          inc a
26464 F5          push af
26465 47          ld b,a
26466 0E 0A       ld c,$0a
26468 CD 38 25    call $2538
26471 CD F1 2B    call $2bf1
26474 1A          ld a,(de)
26475 F5          push af
26476 CD BF 16    call $16bf
26479 F1          pop af
26480 FE 20       cp $20
26482 28 03       jr z,+$03
26484 F1          pop af
26485 18 CA       jr -$36


6777h

26487 F1          pop af
26488 3D          dec a
26489 32 82 67    ld ($6782),a
26492 18 07       jr +$07


677Eh
defb $11 $04 $16 $0a $19 $9e $9f


6785h
26501 3E 02       ld a,$02
26503 CD 01 16    call $1601
26506 11 7E 67    ld de,$677e
26509 01 07 00    ld bc,$0007
26512 CD 3C 20    call $203c


6793h
26515 CD 6A 66    call $666a


6796h
26518 C3 29 61    jp $6129
```

Appendice

Sostituzione delle routine nella ROM

In questa appendice sono rappresentate alcune routine per sostituire quelle originali nella ROM.
Le routine dello Spectrum sono scritte in modo tale che qualunque parametro errato viene evidenziato, filtrato e viene fornito un opportuno messaggio di errore. Se per esempio cercate di PRINT AT 24, 33; "Messaggio", la routine PRINT noterà che linea 24 colonna 33 non appartiene allo schermo. Scrivendo programmi in codice macchina si può ritenere che verranno sempre usati i parametri corretti, ed è quindi possibile, rendendole sempre più veloci, riscrivere le routine nella ROM senza includervi il controllo degli errori, lasciando così al programmatore la massima libertà. I seguenti sei programmi mostrano come riscrivere le routine CLS, ATTR. SET, PRINT, PLOT, UNPLOT, SCR$ e POINT. La routine PRINT gestisce caratteri da 32 a 127 inclusi (anche se può essere adattata per visualizzare qualunque carattere), pone il carattere direttamente sullo schermo rendendo inutile OPEN CHANNEL 2 e permette la visualizzazione sulle linee 22 e 23. La non più usata variabile agli indirizzi 23728/9 viene sfruttata per immagazzinare la posizione di visualizzazione corrente. La routine SCR$, come nella versione BASIC, ritrova i caratteri da 32 a 127 sulle linee da 0 a 21, ma permette il controllo anche delle linee 22 e 23. Le routine PLOT e UNPLOT hanno gli stessi parametri di quelle originali, con 0,0 come angolo inferiore sinistro della linea 21. Se cambiate il valore contenuto all'indirizzo 23774 in 191, potrete controllare l'intero schermo e 0,0 rappresenterà l'angolo inferiore sinistro della linea 23. (Omettendo l'istruzione LD A, 175 agli indirizzi 23774/5, 0,0 rappresenterà l'angolo superiore sinistro della linea 1). Quanto sopra vale anche per la routine POINT. L'indirizzo da modificare è il 23777. Noterete che le rou-

tine PLOT, UNPLOT e POINT usano la stessa subroutine FIND per trovare il pixel da controllare; pertanto, usando le tre routine insieme, potrete condividere questa subroutine, risparmiando memoria. Potrete inoltre aumentare la velocità di esecuzione aggiungendo DI (*disable interrupt*) all'inizio di queste routine (o anche all'inizio del vostro programma) e EI alla fine (o quando necessiti leggere la tastiera tramite KEYSCAN). Con questi programmi come guida potrete scrivere una routine CIRCLE più veloce di quella presente nella ROM dello Spectrum.

Programma A1.1

```
org 42000 23760
CLS ( Lento )

23760 21 00 40     ld hl,16384
L1
23763 36 00        ld (hl),0
23765 23           inc hl
23766 7C           ld a,h
23767 FE 58        cp 88
23769 20 F8        jr nz,L1
23771 C9           ret

CLS ( Veloce )

23772 21 00 40     ld hl,16384
23775 11 01 40     ld de,16385
23778 01 FF 17     ld bc,6143
23781 36 00        ld (hl),0
23783 ED B0        ldir
23785 C9           ret
```

Programma A1.2

```
org 42000 23760
PREDISPOSIZIONE ATTRIBUTE

ESEMPIO
INK 2 PAPER 5
BRIGHT 1 FLASH 1

2+40+64+128=234

23760 3E EA        ld a,234
23762 21 00 58     ld hl,22528
23765 11 01 58     ld de,22529
```

```
23768 01 FF 02     ld bc,767
23771 77           ld (hl),a
23772 ED B0        ldir
23774 C9           ret
```

Programma A 1.3

```
org 42000 23760
ROUTINE VISUALIZZAZIONE STRINGA
PONE IL CARATTERE SULLO SCHERMO
NON USA RST 16d

LA CELLA 23728 CONTIENE IL
NUMERO DELLA COLONNA

LA CELLA 23729 CONTIENE IL
NUMERO DELLA RIGA

23760 11 E4 5C     ld de,DATA
L4
23763 1A           ld a,(de)
23764 CB 7F        bit 7,a
23766 20 06        jr nz,L3
23768 CD F9 5C     call PRINT
23771 13           inc de
23772 18 F5        jr L4
L3

23774 CB BF        res 7,a
23776 CD F9 5C     call PRINT
23779 C9           ret
DATA
defs McGRAW-HILL BOOK Co.

128+CHR 46="."

defb 174

ROUTINE PRINT

RIGA/COLONNA IN HL

PRINT
23801 D5           push de
23802 2A B0 5C     ld hl,(23728)
23805 E5           push hl
23806 F5           push af

CONTROLLA SE FUORI SCHERMO
```

```
23807 AF          xor a
23808 7C          ld a,h
23809 D6 18       sub 24
23811 38 02       jr c,OK
23813 CF          rst 8
defb 4

TROVA LA POSIZIONE DI SCHERMO

OK
23815 11 00 40    ld de,16384
23818 19          add hl,de
23819 7C          ld a,h
23820 E6 07       and 7
23822 0F          rrca
23823 0F          rrca
23824 0F          rrca
23825 B5          or l
23826 6F          ld l,a
23827 3E F8       ld a,248

23829 A4          and h
23830 67          ld h,a

TROVA L'INIZIO DEL CARATTERE
NEL GENERATORE DI CARATTERI

23831 F1          pop af
23832 E5          push hl
23833 11 00 3C    ld de,15360
23836 6F          ld l,a
23837 26 00       ld h,0
23839 29          add hl,hl
23840 29          add hl,hl
23841 29          add hl,hl
23842 19          add hl,de
23843 D1          pop de

PONE IL CARATTERE SULLO SCHERMO

23844 06 08       ld b,8
L1

23846 7E          ld a,(hl)
23847 12          ld (de),a
23848 23          inc hl
23849 14          inc d
23850 10 FA       djnz L1

AGGIORNA LA POSIZIONE
SULLO SCHERMO
```

```
23852 E1          pop hl
23853 2C          inc l
23854 CB 6D       bit 5,l
23856 28 03       jr z,L2
23858 CB AD       res 5,l
23860 24          inc h
L2
23861 22 B0 5C    ld (23728),hl
23864 D1          pop de
23865 C9          ret
```

Programma A1.4

```
org 42000 23760

PROGRAMMA CHE RIMPIAZZA SCREEN$
RICHIEDE H=RIGA, L=COLONNA
RESTITUISCE NEL REGISTRO A IL
CODICE DEL CARATTERE,OPPURE 0
SE NON NE HA TROVATO ALCUNO

23760 26 0A       ld h,10
23762 2E 07       ld l,7
23764 CD D8 5C    call SCR$
23767 C9          ret

ROUTINE SCREEN$

SCR$

CALCOLA L'INDIRIZZO IN MEMORIA
VIDEO


23768 11 00 40    ld de,16384
23771 19          add hl,de
23772 7C          ld a,h
23773 E6 07       and 7
23775 0F          rrca
23776 0F          rrca
23777 0F          rrca
23778 B5          or l
23779 6F          ld l,a
23780 3E F8       ld a,248
23782 A4          and h
23783 67          ld h,a
```

```
CONFRONTA IL CARATTERE SOTTO
ESAME CON QUELLI DEL GENERATO-
RE DI CARATTERI

23784 11 00 3D     ld de,15616
L3
23787 06 08        ld b,8
23789 0E 00        ld c,0

CONTROLLA SE DE=16384
23791 CB 72        bit 6,d
23793 20 11        jr nz,NFOUND
23795 E5           push hl
L2
23796 1A           ld a,(de)
23797 BE           cp (hl)
23798 20 01        jr nz,L1
23800 0C           inc c
L1
23801 13           inc de
23802 24           inc h
23803 10 F7        djnz L2
23805 E1           pop hl
23806 CB 59        bit 3,c
23808 20 04        jr nz,FOUND
23810 18 E7        jr L3
NFOUND
23812 AF           xor a
23813 C9           ret


CONVERTE AL CODICE DI CARATTERE

FOUND
23814 B7           or a
23815 21 00 3C     ld hl,15360
23818 EB           ex de,hl
23819 ED 52        sbc hl,de
23821 06 03        ld b,3
L4
23823 CB 1C        rr h
23825 CB 1D        rr l
23827 10 FA        djnz L4
23829 7D           ld a,l
23830 3D           dec a
23831 C9           ret
```

Programma A1.5

```
org 42000 23760
PROGRAMMA CHE SOSTITUISCE PLOT
RICHIEDE H=y, L=x

23760 26 00        ld h,0
23762 2E 5F        ld l,95
23764 CD D8 5C     call PLOT
23767 C9           ret
PLOT
23768 CD DE 5C     call FIND
23771 B6           or (hl)
23772 77           ld (hl),a
23773 C9           ret

TROVA IL BIT SULLO SCHERMO
RITORNA CON HL=BYTE, A=BIT

FIND
23774 3E AF        ld a,175
23776 94           sub h

23777 67           ld h,a
23778 11 00 40     ld de,16384
23781 7D           ld a,l
23782 E6 07        and 7
23784 F5           push af
23785 7D           ld a,l
23786 E6 F8        and 248
23788 1F           rra
23789 1F           rra
23790 1F           rra
23791 5F           ld e,a
23792 7C           ld a,h
23793 E6 38        and 56
23795 17           rla
23796 17           rla
23797 B3           or e
23798 5F           ld e,a
23799 7C           ld a,h
23800 E6 07        and 7
23802 B2           or d
23803 57           ld d,a

23804 7C           ld a,h
23805 E6 C0        and 192
23807 1F           rra
23808 1F           rra
23809 1F           rra
```

```
23810 B2          or d
23811 57          ld d,a
23812 F1          pop af
23813 2E 80       ld l,128
L1
23815 A7          and a
23816 FE 00       cp 0
23818 28 05       jr z,L2
23820 CB 1D       rr l
23822 3D          dec a
23823 18 F6       jr L1
L2
23825 7D          ld a,l
23826 EB          ex de,hl
23827 C9          ret
```

Programma A1.6

```
org 42000 23760
PROGRAMMA CHE SOSTITUISCE
PLOT OVER 1
RICHIEDE H=y, L=x

23760 26 00       ld h,0
23762 2E 5F       ld l,95
23764 CD D8 5C    call PLOTOVER
23767 C9          ret
PLOTOVER
23768 CD DE 5C    call FIND
23771 AE          xor (hl)
23772 77          ld (hl),a
23773 C9          ret

TROVA IL BIT SULLO SCHERMO
RITORNA CON HL=BYTE, A=BIT

FIND
23774 3E AF       ld a,175

23776 94          sub h
23777 67          ld h,a
23778 11 00 40    ld de,16384
23781 7D          ld a,l
23782 E6 07       and 7
23784 F5          push af
23785 7D          ld a,l
23786 E6 F8       and 248
```

```
23788 1F          rra
23789 1F          rra
23790 1F          rra
23791 5F          ld e,a
23792 7C          ld a,h
23793 E6 38       and 56
23795 17          rla
23796 17          rla
23797 B3          or e
23798 5F          ld e,a
23799 7C          ld a,h
23800 E6 07       and 7
23802 B2          or d

23803 57          ld d,a
23804 7C          ld a,h
23805 E6 C0       and 192
23807 1F          rra
23808 1F          rra
23809 1F          rra
23810 B2          or d
23811 57          ld d,a
23812 F1          pop af
23813 2E 80       ld l,128
L1
23815 A7          and a
23816 FE 00       cp 0
23818 28 05       jr z,L2
23820 CB 1D       rr l
23822 3D          dec a
23823 18 F6       jr L1
L2
23825 7D          ld a,l
23826 EB          ex de,hl
23827 C9          ret
```

Programma A1.7

```
org 42000 23760
PROGRAMMA CHE SOSTITUISCE POINT
RICHIEDE H=y, L=x

23760 26 00        ld h,0
23762 2E 5F        ld l,95
23764 CD D8 5C     call POINT
23767 C9           ret
POINT
23768 CD E0 5C     call FIND
23771 A6           and (hl)
23772 C8           ret z
23773 3E 01        ld a,1
23775 C9           ret

TROVA IL BIT SULLO SCHERMO
RITORNA CON HL=BYTE, A=BIT

FIND
23776 3E AF        ld a,175

23778 94           sub h
23779 67           ld h,a
23780 11 00 40     ld de,16384
23783 7D           ld a,l
23784 E6 07        and 7
23786 F5           push af
23787 7D           ld a,l
23788 E6 F8        and 248
23790 1F           rra
23791 1F           rra
23792 1F           rra
23793 5F           ld e,a
23794 7C           ld a,h
23795 E6 38        and 56
23797 17           rla
23798 17           rla
23799 B3           or e
23800 5F           ld e,a
23801 7C           ld a,h
23802 E6 07        and 7
23804 B2           or d

23805 57           ld d,a
23806 7C           ld a,h
23807 E6 C0        and 192
23809 1F           rra
```

```
23810 1F          rra
23811 1F          rra
23812 B2          or d
23813 57          ld d,a
23814 F1          pop af
23815 2E 80       ld l,128
L1
23817 A7          and a
23818 FE 00       cp 0
23820 28 05       jr z,L2
23822 CB 1D       rr l
23824 3D          dec a
23825 18 F6       jr L1
L2
23827 7D          ld a,l
23828 EB          ex de,hl
23829 C9          ret
```

Appendice

I codici macchina dello Z80

B

Tabella B.1 Codici macchina (mnemonici) dello Z80

(HL)–è il numero contenuto nella locazione il cui indirizzo è puntato da HL
NN–supponendo che sia scritto come N_1N_2, N_2 è inserito nel byte basso (da 0 a 255), seguito da N_1 nel byte alto (da 0 a 255 per 256). Per esempio, il numero decimale 4000 è inserito come $3840+160=160$ (N_2), 15 (N_1), poiché $15\times256=3840$.
Registro A–è l'accumulatore.

Decimale	*Byte*	*Es.*	*Codifica*	*Descrizione*
0	1	00	nop	Nessuna operazione
1	3	01	ld bc,NN	Carica nella coppia di registri BC il valore NN
2	1	02	ld (bc),a	Immagazzina l'accumulatore nella locazione di memoria indirizzata dalla coppia di registri BC
3	1	03	inc bc	Incrementa di 1 il contenuto della coppia di registri BC
4	1	04	inc b	Incrementa di 1 il contenuto del registro B
5	1	05	dec b	Diminuisce di 1 il contenuto del registro B
6	2	06	ld b,N	Carica nel registro B il valore N
7	1	07	rlca	Rotazione circolare verso sinistra dell'accumulatore
8	1	08	ex af, af	Scambia il contenuto di A e di F con quello di A' e F'
9	1	09	add hl,bc	Somma il contenuto della coppia di registri HL con quello di BC, lasciando il risultato in HL
10	3	0A	ld a,(bc)	Immagazzina nell'accumulatore il contenuto della locazione di memoria indirizzata dalla coppia di registri BC
11	1	0B	dec bc	Diminuisce di 1 il contenuto della coppia di registri BC
12	1	0C	dec c	Incrementa di 1 il contenuto del registro C
13	1	0D	dec c	Diminuisce di 1 il contenuto del registro C
14	2	0E	ld c,N	Carica nel registro C il valore N
15	1	0F	rrca	Rotazione circolare dell'accumulatore verso destra
16	2	10	djnz x	Diminuisce di 1 il contenuto del registro B, e, se il risultato è zero, salta in avanti o indietro di x
17	3	11	ld de,NN	Carica nella coppia di registri DE il valore NN
18	1	12	ld(de),a	Immagazzina l'accumulatore nella locazione di memoria indirizzata dalla coppia di registri DE

Decimale	*Byte*	*Es.*	*Codifica*	*Descrizione*
19	1	13	inc de	Incrementa di 1 il contenuto della coppia di registri DE
20	1	14	inc d	Incrementa di 1 il contenuto del registro D
21	1	15	dec d	Diminuisce di 1 il contenuto del registro D
22	2	16	ld d,N	Carica nel registro D il valore N
23	1	17	rla	Rotazione circolare verso sinistra dell'accumulatore attraverso il flag di carry (riporto)
24	2	18	jr x	Salta in avanti o in indietro di x
25	1	19	add hl,de	Somma il contenuto della coppia di registri HL con quello di DE, lasciando il risultato in HL
26	3	1A	ld a,(de)	Immagazzina nell'accumulatore il contenuto della locazione di memoria indirizzata dalla coppia di registri DE
27	1	1B	dec de	Diminuisce di 1 il contenuto della coppia di registri DE
28	1	1C	inc e	Incrementa di 1 il contenuto del registro E
29	1	1D	dec e	Diminuisce di 1 il contenuto del registro E
30	2	1E	ld e,N	Carica nel registro E il valore N
31	1	1F	rra	Rotazione circolare verso destra dell'accumulatore attraverso il flag di carry
32	2	20	jr nz, x	Salta in avanti o in indietro di x, se il flag Z è a zero (cioè, se è non-zero)
33	3	21	ld hl,NN	Carica nella coppia di registri HL il valore NN
34	3	22	ld (NN),hl	Immagazzina il contenuto della coppia di registri HL nella locazione NN
35	1	23	inc hl	Incrementa di 1 il contenuto della coppia di registri HL
36	1	24	inc h	Incrementa di 1 il contenuto del registro H
37	1	25	dec h	Diminuisce di 1 il contenuto del registro H
38	2	26	ld h,N	Carica nel registro H il valore N
39	1	27	daa	Aggiustamento decimale dell'accumulatore
40	2	28	jr z, x	Salta in avanti o in indietro di x, se il flag Z è a uno (cioè se è zero)
41	1	29	add hl,hl	Raddoppia il contenuto della coppia di registri HL
42	3	2A	ld hl,(NN)	Immagazzina nella coppia di registri HL il contenuto della locazione NN
43	1	2B	dec hl	Diminuisce di 1 il contenuto della coppia di registri HL
44	1	2C	inc l	Incrementa di 1 il contenuto del registro L
45	1	2D	dec l	Diminuisce di 1 il contenuto del registro L
46	2	2E	ld l,N	Carica nel registro L il valore N
47	1	2F	cpl	Complementa l'accumulatore
48	2	30	jr nc, x	Salta in avanti o in indietro di x se il flag C è a zero (cioè se è non-carry)
49	3	31	ld sp,NN	Carica nel puntatore allo stack (stack pointer) il valore NN
50	3	32	ld (NN),a	Immagazzina l'accumulatore nella locazione NN
51	1	33	inc sp	Incrementa di 1 il contenuto dello stack pointer
52	1	34	inc (hl)	Incrementa di 1 il contenuto della locazione di memoria indirizzata dalla coppia di registri HL
53	1	35	dec (hl)	Diminuisce di 1 il contenuto della locazione di memoria indirizzata dalla coppia di registri HL
54	2	36	ld (hl),N	Carica il valore N nella locazione di memoria indirizzata dalla coppia di registri HL
55	1	37	scf	Pone a uno il flag C
56	2	38	jr c, x	Salta in avanti o in indietro di x se il flag C è a uno (cioè se è carry)
57	1	39	add hl,sp	Somma al contenuto della coppia di registri HL il contenuto del registro SP (stack pointer) lasciando il risultato in HL
58	3	3A	ld a,(NN)	Immagazzina nell'accumulatore il contenuto della locazione NN
59	1	3B	dec sp	Diminuisce di 1 il contenuto del registro SP (stack pointer)
60	1	3C	inc a	Incrementa di 1 il contenuto dell'accumulatore

Decimale	*Byte*	*Es.*	*Codifica*	*Descrizione*
61	1	3D	dec a	Diminuisce di 1 il contenuto dell'accumulatore
62	2	3E	ld a,N	Immagazzina nell'accumulatore il valore N
63	1	3F	ccf	Completa il flag C
64	1	40	ld b,b	Copia il contenuto del registro B nel registro B
65	1	41	ld b,c	Copia il contenuto del registro C nel registro B
66	1	42	ld b,d	Copia il contenuto del registro D nel registro B
67	1	43	ld b,e	Copia il contenuto del registro E nel registro B
68	1	44	ld b,h	Copia il contenuto del registro H nel registro B
69	1	45	ld b,l	Copia il contenuto del registro L nel registro B
70	1	46	ld b,(hl)	Carica il registro B dalla locazione di memoria indirizzata dalla coppia di registri HL
71	1	47	ld b,a	Copia il contenuto dell'accumulatore nel registro B
72	1	48	ld c,b	Copia il contenuto del registro B nel registro C
73	1	49	ld c,c	Copia il contenuto del registro C nel registro C
74	1	4A	ld c,d	Copia il contenuto del registro D nel registro C
75	1	4B	ld c,e	Copia il contenuto del registro E nel registro C
76	1	4C	ld c,h	Copia il contenuto del registro H nel registro C
77	1	4D	ld c,l	Copia il contenuto del registro L nel registro C
78	1	4E	ld c,(hl)	Carica il registro C dalla locazione di memoria indirizzata dalla coppia di registri HL
79	1	4F	ld c,a	Copia il contenuto dell'accumulatore nel registro C
80	1	50	ld d,b	Copia il contenuto del registro B nel registro D
81	1	51	ld d,c	Copia il contenuto del registro C nel registro D
82	1	52	ld d,d	Copia il contenuto del registro D nel registro D
83	1	53	ld d,e	Copia il contenuto del registro E nel registro D
84	1	54	ld d,h	Copia il contenuto del registro H nel registro D
85	1	55	ld d,l	Copia il contenuto del registro L nel registro D
86	1	56	ld d,(hl)	Carica il registro D dalla locazione di memoria indirizzata dalla coppia di registri HL
87	1	57	ld d,a	Copia il contenuto dell'accumulatore nel registro D
88	1	58	ld e,b	Copia il contenuto del registro B nel registro E
89	1	59	ld e,c	Copia il contenuto del registro C nel registro E
90	1	5A	ld e,d	Copia il contenuto del registro D nel registro E
91	1	5B	ld e,e	Copia il contenuto del registro E nel registro E
92	1	5C	ld e,h	Copia il contenuto del registro H nel registro E
93	1	5D	ld e,l	Copia il contenuto del registro L nel registro E
94	1	5E	ld e,(hl)	Carica il registro E dalla locazione di memoria indirizzata dalla coppia di registri HL
95	1	5F	ld e,a	Copia il contenuto dell'accumulatore nel registro E
96	1	60	ld h,b	Copia il contenuto del registro B nel registro H
97	1	61	ld h,c	Copia il contenuto del registro C nel registro H
98	1	62	ld h,d	Copia il contenuto del registro D nel registro H
99	1	63	ld h,e	Copia il contenuto del registro E nel registro H
100	1	64	ld h,h	Copia il contenuto del registro H nel registro H
101	1	65	ld h,l	Copia il contenuto del registro L nel registro H
102	1	66	ld h,(hl)	Carica il registro H dalla locazione di memoria indirizzata dalla coppia di registri HL
103	1	67	ld h,a	Copia il contenuto dell'accumulatore nel registro H
104	1	68	ld l,b	Copia il contenuto del registro B nel registro L
105	1	69	ld l,c	Copia il contenuto del registro C nel registro L
106	1	6A	ld l,d	Copia il contenuto del registro D nel registro L
107	1	6B	ld l,e	Copia il contenuto del registro E nel registro L
108	1	6C	ld l,h	Copia il contenuto del registro H nel registro L
109	1	6D	ld l,l	Copia il contenuto del registro L nel registro L
110	1	6E	ld l,(hl)	Carica il registro L dalla locazione di memoria indirizzata dalla coppia di registri HL
111	1	6F	ld l,a	Copia il contenuto dell'accumulatore nel registro L

Decimale	*Byte*	*Es.*	*Codifica*	*Descrizione*
112	1	70	ld (hl),b	Copia il contenuto del registro B nella locazione indirizzata da HL
113	1	71	ld (hl),c	Copia il contenuto del registro C nella locazione indirizzata da HL
114	1	72	ld (hl),d	Copia il contenuto del registro D nella locazione indirizzata da HL
115	1	73	ld (hl),e	Copia il contenuto del registro E nella locazione indirizzata da HL
116	1	74	ld (hl),h	Copia il contenuto del registro H nella locazione indirizzata da HL
117	1	75	ld (hl),l	Copia il contenuto del registro H nella locazione indirizzata da HL
118	1	76	halt	Blocca la CPU
119	1	77	ld (hl),a	Copia il contenuto dell'accumulatore nella locazione indirizzata da HL
120	1	78	ld a,b	Copia il contenuto del registro B nell'accumulatore
121	1	79	ld a,c	Copia il contenuto del registro C nell'accumulatore
122	1	7A	ld a,d	Copia il contenuto del registro D nell'accumulatore
123	1	7B	ld a,e	Copia il contenuto del registro E nell'accumulatore
124	1	7C	ld a,h	Copia il contenuto del registro H nell'accumulatore
125	1	7D	ld a,l	Copia il contenuto del registro L nell'accumulatore
126	1	7E	ld a,(hl)	Carica l'accumulatore dalla locazione di memoria indirizzata dalla coppia di registri HL
127	1	7F	ld a,a	Copia il contenuto dell'accumulatore nell'accumulatore
128	1	80	add a,b	B+A→A
129	1	81	add a,c	C+A→A
130	1	82	add a,d	D+A→A
131	1	83	add a,e	E+A→A
132	1	84	add a,h	H+A→A
133	1	85	add a,l	L+A→A
134	1	86	add a,(hl)	(HL)+A→A
135	1	87	add a,a	A+A→A
136	1	88	adc a,b	B+A+carry→A
137	1	89	adc a,c	C+A+carry→A
138	1	8A	adc a,d	D+A+carry→A
139	1	8B	adc a,e	E+A+carry→A
140	1	8C	adc a,h	H+A+carry→A
141	1	8D	adc a,l	L+A+carry→A
142	1	8E	adc a,(hl)	(HL)+A+carry→A
143	1	8F	adc a,a	A+A+carry→A
144	1	90	sub b	A−B→A
145	1	91	sub c	A−C→A
146	1	92	sub d	A−D→A
147	1	93	sub e	A−E→A
148	1	94	sub h	A−H→A
149	1	95	sub l	A−L→A
150	1	96	sub (hl)	A−(HL)→A
151	1	97	sub a	A−A→A
152	1	98	sbc a,b	A−B−carry→A
153	1	99	sbc a,c	A−C−carry→A
154	1	9A	sbc a,d	A−D−carry→A
155	1	9B	sbc a,e	A−E−carry→A
156	1	9C	sbc a,h	A−H−carry→A
157	1	9D	sbc a,l	A−L−carry→A
158	1	9E	sbc a,(hl)	A−(HL)−carry→A
159	1	9F	sbc a,a	A−A−carry→A
160	1	A0	and b	A and B→A

Decimale	*Byte*	*Es.*	*Codifica*	*Descrizione*
161	1	A1	and c	A and C→A
162	1	A2	and d	A and D→A
163	1	A3	and e	A and E→A
164	1	A4	and h	A and H→A
165	1	A5	and l	A and L→A
166	1	A6	and (hl)	A and (HL)→A
167	1	A7	and a	A and A→A
168	1	A8	xor b	A exclusive or B→A
169	1	A9	xor c	A exclusive or C→A
170	1	AA	xor d	A exclusive or D→A
171	1	AB	xor e	A exclusive or E→A
172	1	AC	xor h	A exclusive or H→A
173	1	AD	xor l	A exclusive or L→A
174	1	AE	xor (hl)	A exclusive or (HL)→A
175	1	AF	xor a	A exclusive or A→A
176	1	B0	or b	A or B→A
177	1	B1	or c	A or C→A
178	1	B2	or d	A or D→A
179	1	B3	or e	A or E→A
180	1	B4	or h	A or H→A
181	1	B5	or l	A or L→A
182	1	B6	or(hl)	A or (HL)→A
183	1	B7	or a	A or A→A
184	1	B8	cp b	Sottrae il contenuto del registro B dall'accumulatore, scartando il risultato
185	1	B9	cp c	Sottrae il contenuto del registro C dall'accumulatore, scartando il risultato
186	1	BA	cp d	Sottrae il contenuto del registro D dall'accumulatore, scartando il risultato
187	1	BB	cp e	Sottrae il contenuto del registro E dall'accumulatore, scartando il risultato
188	1	BC	cp h	Sottrae il contenuto del registro H dall'accumulatore, scartando il risultato
189	1	BD	cp l	Sottrae il contenuto del registro L dall'accumulatore, scartando il risultato
190	1	BE	cp (hl)	Sottrae dall'accumulatore il contenuto della locazione indirizzata dalla coppia di registri HL, scartando il risultato (cioè lasciando invariato il contenuto dell'accumulatore)
191	1	BF	cp a	Sottrae il contenuto dell'accumulatore, scartando il risultato
192	1	C0	ret nz	Ritorna se il flag Z è a zero (cioè se è non-zero)
193	1	C1	pop bc	Carica dallo stack la coppia di registri BC
194	3	C2	jp nz,NN	Salta alla locazione NN se il flag Z è a zero
195	3	C3	jp NN	Salta alla locazione NN
196	3	C4	call nz,NN	Chiama la routine alla locazione NN se il flag Z è a zero
197	1	C5	push bc	Salva nello stack il contenuto della coppia di registri BC
198	2	C6	add a,N	A+N→A
199	1	C7	rst 0	Chiama la routine iniziante alla locazione 0000
200	1	C8	ret z	Ritorna se il flag Z è a uno (cioè se è zero)
201	1	C9	ret	Ritorna
202	3	CA	jp z,NN	Salta alla locazione NN se il flag Z è a uno
203		CB		Vedi l'insieme di routine CB
204	3	CC	call z,NN	Chiama la routine alla locazione NN se il flag Z è a uno
205	3	CD	call NN	Chiama la routine alla locazione NN
207	1	CF	rst 8	Chiama la routine di errore all'indirizzo 0008
208	1	D0	ret nc	Ritorna se il flag C è a zero (cioè se non-carry)
209	1	D1	pop de	Carica dallo stack la coppia di registri DE
210	3	D2	jp nc,NN	Salta alla locazione NN se il flag C è a zero

Decimale	*Byte*	*Es.*	*Codifica*	*Descrizione*
211	2	D3	out(N),a	Emette il contenuto dell'accumulatore al port N
212	3	D4	call nc,NN	Chiama la routine alla locazione NN se il flag C è a zero
213	1	D5	push de	Salva nello stack il contenuto della coppia di registri DE
214	2	D6	sub N	A−N→A
215	1	D7	rst 16	Chiama la routine di stampa all'indirizzo 0010
216	1	D8	ret c	Ritorna se il flag C è a uno
217	1	D9	exx	Scambia i registri principali con quelli alternativi
218	3	DA	jp c,NN	Salta alla locazione NN se il flag C è a uno
219	2	DB	in a,(N)	Carica l'accumulatore dal port N
220	3	DC	call c,NN	Chiama la routine alla locazione NN se il flag C è a uno
221		DD	[prefissi delle istruzioni che usano ix]	Vedi Tabella A.2
222	2	DE	sbc a,N	A−N−carry→A
223	1	DF	rst 24	Chiama la routine di carattere all'indirizzo 0018
224	1	E0	ret po	Ritorna se il flag P/O è a zero
225	1	E1	pop hl	Carica dallo stack la coppia di registri HL
226	3	E2	jp po,NN	Salta alla locazione NN se il flag P/O è a zero
227	1	E3	ex(sp),hl	Scambia il contenuto del registro SP con quello della coppia di registri HL
228	3	E4	call po,NN	Chiama la routine all'indirizzo NN se il flag P/O è a zero
229	1	E5	push hl	Salva nello stack il contenuto della coppia di registri HL
230	2	E6	and N	A and N→A
231	1	E7	rst 32	Chiama la routine di carattere alla locazione 0020
232	1	E8	ret pe	Ritorna se il flag P/O è a uno
233	1	E9	jp (hl)	Salta alla locazione indirizzata dalla coppia di registri HL
234	3	EA	jp pe,NN	Salta alla locazione NN se il flag P/O è a uno
235	1	EB	ex de,hl	Scambia il contenuto della coppia di registri DE con quello della coppia HL
236	3	EC	call pe,NN	Chiama la routine all'indirizzo NN se il flag P/O è a uno
237		ED		Vedi l'insieme di routine ED
238	2	EE	xor N	A exclusive or N→A
239	1	EF	rst 40	Chiama la routine calcolatore all'indirizzo 0028
240	1	F0	ret p	Ritorna se il flag P è a zero
241	1	F1	pop af	Carica dallo stack la coppia di registri AF
242	3	F2	jp p,NN	Salta alla locazione NN se il flag P è a zero
243	1	F3	di	Disabilita gli interrupt
244	3	F4	call p,NN	Chiama la routine all'indirizzo NN se il flag P è a zero
245	1	F5	push af	Salva nello stack il contenuto della coppia di registri AF
246	2	F6	or N	A or N→A
247	1	F7	rst 48	Chiama la routine spazio di lavoro all'indirizzo 0030
248	1	F8	ret m	Ritorna se il flag P è a uno
249	1	F9	ld sp,hl	Copia il contenuto del registro HL nel registro SP (stack pointer)
250	3	FA	jp m,NN	Salta alla locazione NN se il flag P è a uno
251	1	FB	ei	Abilita gli interrupt
252	3	FC	call m,NN	Chiama la routine all'indirizzo NN se il flag P è a uno
253		FD	[prefissi delle istruzioni che usano iy]	Vedi Tabella A.2
254	2	FE	cp N	Sottrae al contenuto dell'accumulatore il valore N scaricando il risultato
255	1	FF	rst 56	Chiama la routine all'indirizzo 0038

Tabella B.2 Codici utilizzati con DD e FD

d – significa spostamento (displacement): dd e dddd significano rispettivamente numeri di uno e due byte.
La tabella si riferisce ai mnemonici DD (221 decimale) in relazione col registro IX.
I mnemonici FD (253 decimale) usano operazioni simili, ma in relazione col registro IY.

DD 09	ADD	IX,BC	DD CB d 06	RLC	(IX+d)
DD 19	ADD	IX,DE	DD CB d 0E	RRC	(IX+d)
DD 21 +dddd	LD	IX,+dddd	DD CB d 16	RL	(IX+d)
DD 22 addr	LD	(addr),IX	DD CB d 1E	RR	(IX+d)
DD 23	INC	IX	DD CB d 26	SLA	(IX+d)
DD 29	ADD	IX,IX	DD CB d 2E	SRA	(IX+d)
DD 2A addr	LD	IX,(addr)	DD CB d 3E	SRL	(IX+d)
DD 2B	DEC	IX	DD CB d 46	BIT	0,(IX+d)
DD 34 d	INC	(IX+d)	DD CB d 4E	BIT	1,(IX+d)
DD 35 d	DEC	(IX+d)	DD CB d 56	BIT	2,(IX+d)
DD 36 d +dd	LD	(IX+d),+dd	DD CB d 5E	BIT	3,(IX+d)
DD 39	ADD	IX,SP	DD CB d 66	BIT	4,(IX+d)
DD 46 d	LD	B,(IX+d)	DD CB d 6E	BIT	5,(IX+d)
DD 4E d	LD	C,(IX+d)	DD CB d 76	BIT	6,(IX+d)
DD 56 d	LD	D,(IX+d)	DD CB d 7E	BIT	7,(IX+d)
DD 5E d	LD	E,(IX+d)	DD CB d 86	RES	0,(IX+d)
DD 66 d	LD	H,(IX+d)	DD CB d 8E	RES	1,(IX+d)
DD 6E d	LD	L,(IX+d)	DD CB d 96	RES	2,(IX+d)
DD 70 d	LD	(IX+d),B	DD CB d 9E	RS	3,(IX+d)
DD 71 d	LD	(IX+d),C	DD CB d A6	RES	4,(IX+d)
DD 72 d	LD	(IX+d),D	DD CB d AE	RES	5,(IX+d)
DD 73 d	LD	(IX+d),E	DD CB d B6	RES	6,(IX+d)
DD 74 d	LD	(IX+d),H	DD CB d BE	RES	7,(IX+d)
DD 75 d	LD	(IX+d),L	DD CB d C6	SET	0,(IX+d)
DD 77 d	LD	(IX+d),A	DD CB d CE	SET	1,(IX+d)
DD 7E d	LD	A,(IX+d)	DD CB d D6	SET	2,(IX+d)
	ADD	A,(IX+d)	DD CB d DE	SET	3,(IX+d)
DD 8E d	ADC	A,(IX+d)	DD CB d E6	SET	4,(IX+d)
DD 96 d	SUB	(IX+d)	DD CB d EE	SET	5,(IX+d)
DD 9E d	SBC	A,(IX+d)	DD CB d F6	SET	6,(IX+d)
DD A6 d	AND	(IX+d)	DD CB d FE	SET	7,(IX+d)
DD AE d	XOR	(IX+d)	DD E1	POP	IX
DD B6 d	OR	(IX+d)	DD E3	EX	(SP),IX
DD BE d	CP	(IX+d)	DD E5	PUSH	IX
			DD E9	JP	(IX)
			DD F9	LD	SP,IX

Tabella B.3 Istruzioni ED (237 decimale)

ED 40 IN B,(C)	ED 50 IN D,(C)	ED 60 IN H,(C)		ED A0 LDI	ED B0 LDIR
ED 41 OUT (C),B	ED 51 OUT (C),D	ED 61 OUT (C),H		ED A1 CPI	ED B1 CPIR
ED 42 SBC HL,BC	ED 52 SBC HL,DE	ED 62 SBC HL,HL	ED 72 SBC HL,SP	ED A2 INI	ED B2 INIR
ED 43 (addr),BC	ED 53 (addr),DE	ED 63 (addr),HL	ED 73 (addr),SP	ED A3 OUTI	ED B3 OTIR
ED 44 NEG					
ED 45 RETN					
ED 46 IM 0	ED 56 IM 1	ED 66 IM 2			
ED 47 LD I,A	ED 57 LD A,I	ED 67 RRD			
ED 48 IN C,(C)	ED 58 IN E,(C)	ED 68 IN L,(C)	ED 78 IN A,(C)	ED A8 LDD	ED B8 LDDR
ED 49 OUT (C),C	ED 59 OUT (C),E	ED 69 OUT (C),L	ED 79 OUT (C),A	ED A9 CPD	ED B9 CPDR
ED 4A ADC HL,BC	ED 5A ADC HL,DE	ED 6A ADC HL,HL	ED 7A ADC HL,SP	ED AA IND	ED BA INDR
ED 4B LD BC,(addr)	ED 5B LD DE,(addr)	ED 6B LD HL,(addr)	ED 7B LD SP,(addr)	ED AB OUTD	ED BB OTRD
ED 4D RETI					
ED 4F LD R,A	ED 5F LD A,R	ED 6F RLD			

Appendice

Tabelle di conversione esadecimali-decimali

C

Le due tabelle riportano la conversione tra valori decimali ed esadecimali, rispettivamente per i byte bassi (valori decimali da 0 a 255) e per i byte alti (valori decimali da 0 a 65280), corrispondenti a valori esadecimali da 00 a FF.
Nella prima, per valori decimali superiori a 127, viene fornito anche il complemento a due.

Tabella C.1 Decimali da 0 a 255 — Esadecimali da 00 a FF, byte basso – Complemento a 2

Dec.	Esad.	Dec.	Esad.	Dec.	Esad.	C.a2	Dec.	Esad.	C.a 2
0	00	64	40	128	80	−128	192	C0	−64
1	01	65	41	129	81	−127	193	C1	−63
2	02	66	42	130	82	−126	194	C2	−62
3	03	67	43	131	83	−125	195	C3	−61
4	04	68	44	132	84	−124	196	C4	−60
5	05	69	45	133	85	−123	197	C5	−59
6	06	70	46	134	86	−122	198	C6	−58
7	07	71	47	135	87	−121	199	C7	−57
8	08	72	48	136	88	−120	200	C8	−56
9	09	73	49	137	89	−119	201	C9	−55
10	0A	74	4A	138	8A	−118	202	CA	−54
11	0B	75	4B	139	8B	−117	203	CB	−53
12	0C	76	4C	140	8C	−116	204	CC	−52
13	0D	77	4D	141	8D	−115	205	CD	−51
14	0E	78	4E	142	8E	−114	206	CE	−50
15	0F	79	4F	143	8F	−113	207	CF	−49
16	10	80	50	144	90	−112	208	D0	−48
17	11	81	51	145	91	−111	209	D1	−47
18	12	82	52	146	92	−110	210	D2	−46
19	13	83	53	147	93	−109	211	D3	−45
20	14	84	54	148	94	−108	212	D4	−44
21	15	85	55	149	95	−107	213	D5	−43
22	16	86	56	150	96	−106	214	D6	−42
23	17	87	57	151	97	−105	215	D7	−41
24	18	88	58	152	98	−104	216	D8	−40
25	19	89	59	153	99	−103	217	D9	−39
26	1A	90	5A	154	9A	−102	218	DA	−38
27	1B	91	5B	155	9B	−101	219	DB	−37
28	1C	92	5C	156	9C	−100	220	DC	−36
29	1D	93	5D	157	9D	−99	221	DD	−35
30	1E	94	5E	158	9E	−98	222	DE	−34
31	1F	95	5F	159	9F	−97	223	DF	−33
32	20	96	60	160	A0	−96	224	E0	−32
33	21	97	61	161	A1	−95	225	E1	−31
34	22	98	62	162	A2	−94	226	E2	−30
35	23	99	63	163	A3	−93	227	E3	−29
36	24	100	64	164	A4	−92	228	E4	−28
37	25	101	65	165	A5	−91	229	E5	−27
38	26	102	66	166	A6	−90	230	E6	−26
39	27	103	67	167	A7	−89	231	E7	−25
40	28	104	68	168	A8	−88	232	E8	−24
41	29	105	69	169	A9	−87	233	E9	−23
42	2A	106	6A	170	AA	−86	234	EA	−22
43	2B	107	6B	171	AB	−85	235	EB	−21
44	2C	108	6C	172	AC	−84	236	EC	−20
45	2D	109	6D	173	AD	−83	237	ED	−19
46	2E	110	6E	174	AE	−82	238	EE	−18
47	2F	111	6F	175	AF	−81	239	EF	−17
48	30	112	70	176	B0	−80	240	F0	−16
49	31	113	71	177	B1	−79	241	F1	−15
50	32	114	72	178	B2	−78	242	F2	−14
51	33	115	73	179	B3	−77	243	F3	−13
52	34	116	74	180	B4	−76	244	F4	−12
53	35	117	75	181	B5	−75	245	F5	−11
54	36	118	76	182	B6	−74	246	F6	−10
55	37	119	77	183	B7	−73	247	F7	−9
56	38	120	78	184	B8	−72	248	F8	−8
57	39	121	79	185	B9	−71	249	F9	−7
58	3A	122	7A	186	BA	−70	250	FA	−6
59	3B	123	7B	187	BB	−69	251	FB	−5
60	3C	124	7C	188	BC	−68	252	FC	−4
61	3D	125	7D	189	BD	−67	253	FD	−3
62	3E	126	7E	190	BE	−66	254	FE	−2
63	3F	127	7F	191	BF	−65	255	FF	−1

Tabella C.2 Decimali da 0 a 54280 — Esadecimali da 00 a FF, byte alto

Decimali	Esad.	Decimali	Esad.	Decimali	Esad.	Decimali	Esad.
0	00	16 384	40	32 768	80	49 152	C0
256	01	16 640	41	33 024	81	49 408	C1
512	02	16 896	42	33 280	82	49 664	C2
768	03	17 152	43	33 536	83	49 920	C3
1 024	04	17 408	44	33 792	84	50 176	C4
1 280	05	17 664	45	34 048	85	50 432	C5
1 536	06	17 920	46	34 304	86	50 688	C6
1 792	07	18 176	47	34 560	87	50 944	C7
2 048	08	18 432	48	34 816	88	51 200	C8
2 304	09	18 688	49	35 072	89	51 456	C9
2 560	0A	18 944	4A	35 328	8A	51 712	CA
2 816	0B	19 200	4B	35 584	8B	51 968	CB
3 072	0C	19 456	4C	35 840	8C	52 224	CC
3 328	0D	19 712	4D	36 096	8D	52 480	CD
3 584	0E	19 968	4E	36 352	8E	52 736	CE
3 840	0F	20 224	4F	36 608	8F	52 992	CF
4 096	10	20 480	50	36 864	90	53 248	D0
4 352	11	20 736	51	37 120	91	53 504	D1
4 608	12	20 992	52	37 376	92	53 760	D2
4 864	13	21 248	53	37 632	93	54 016	D3
5 120	14	21 504	54	37 888	94	54 272	D4
5 376	15	21 760	55	38 144	95	54 528	D5
5 632	16	22 016	56	38 400	96	54 784	D6
5 888	17	22 272	57	38 656	97	55 040	D7
6 144	18	22 528	58	38 912	98	55 296	D8
6 400	19	22 784	59	39 168	99	55 552	D9
6 656	1A	23 040	5A	39 424	9A	55 808	DA
6 912	1B	23 296	5B	39 680	9B	56 064	DB
7 168	1C	23 552	5C	39 936	9C	56 320	DC
7 424	1D	23 808	5D	40 192	9D	56 576	DD
7 680	1E	24 064	5E	40 448	9E	56 832	DE
7 936	1F	24 320	5F	40 704	9F	57 088	DF
8 192	20	24 576	60	40 960	A0	57 344	E0
8 448	21	24 832	61	41 216	A1	57 600	E1
8 704	22	25 088	62	41 472	A2	57 856	E2
8 960	23	25 344	63	41 728	A3	58 112	E3
9 216	24	25 600	64	41 984	A4	58 368	E4
9 472	25	25 856	65	42 240	A5	58 624	E5
9 728	26	26 112	66	42 496	A6	58 880	E6
9 984	27	26 368	67	42 752	A7	59 136	E7
10 240	28	26 624	68	43 008	A8	59 392	E8
10 496	29	26 880	69	43 264	A9	59 648	E9
10 752	2A	27 136	6A	43 520	AA	59 904	EA
11 008	2B	27 392	6B	43 776	AB	60 160	EB
11 264	2C	27 648	6C	44 032	AC	60 416	EC
11 520	2D	27 904	6D	44 288	AD	60 672	ED
11 776	2E	28 160	6E	44 544	AE	60 928	EE
12 032	2F	28 416	6F	44 800	AF	61 184	EF
12 288	30	28 672	70	45 056	B0	61 440	F0
12 544	31	28 928	71	45 312	B1	61 696	F1
12 800	32	29 184	72	45 568	B2	61 952	F2
13 056	33	29 440	73	45 824	B3	62 208	F3
13 312	34	29 696	74	46 080	B4	62 464	F4
13 568	35	29 952	75	46 336	B5	62 720	F5
13 824	36	30 208	76	46 592	B6	62 976	F6
14 080	37	30 464	77	46 848	B7	63 232	F7
14 336	38	30 720	78	47 104	B8	63 488	F8
14 592	39	30 976	79	47 360	B9	63 744	F9
14 848	3A	31 232	7A	47 616	BA	64 000	FA
15 104	3B	31 488	7B	47 872	BB	64 256	FB
15 360	3C	31 744	7C	48 128	BC	64 512	FC
15 616	3D	32 000	7D	48 384	BD	64 768	FD
15 872	3E	32 256	7E	48 640	BE	65 024	FE
16 128	3F	32 512	7F	48 896	BF	65 280	FF

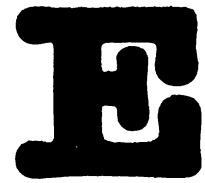
Appendice

Literal D

Literal d	Nome	Azione
1	EXCHANGE	Scambia i due valori in cima allo stack
3	SUBTRACT	Sottrae il valore in cima allo stack da quello sottostante
4	MULTIPLY	Moltiplica i due valori in cima allo stack
5	DIVIDE	Divide il valore in cima allo stack per quello sottostante
15	ADD	Somma i due valori in cima allo stack
31	SIN	Seno del valore in cima allo stack
32	COS	Coseno del valore in cima allo stack
33	TAN	Tangente del valore in cima allo stack
39	INT	Parte intera del valore in cima allo stack
40	SQR	Radice quadrata del valore in cima allo stack
49	DUPLICATE	Duplica in cima allo stack il valore ivi presente
52	STK	Immagazzina nello stack dati in forma compressa (vedi Capitolo 7)
56	END	Fine della routine di calcolo e ritorno al programma in codice macchina
61	RESTACK	Sostituisce al valore in cima allo stack la sua rappresentazione in virgola mobile
160	0	Stack 0
161	1	Stack 1
162	1/2	Stack 1/2
163	π/2	Stack π/2
164	10	Stack 10

Appendice

Mappa della memoria del video



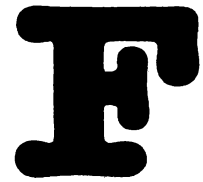
Sfruttando questa tabella potrete agevolmente trovare le celle di memoria corrispondenti alle varie zone del video.
Esempio: il byte della quinta riga del carattere in linea 13, colonna 27 è dato da

indirizzo base riga 5, linea 13	4BA0+
numero colonna	1B=
indirizzo del byte corrispondente	4BBB

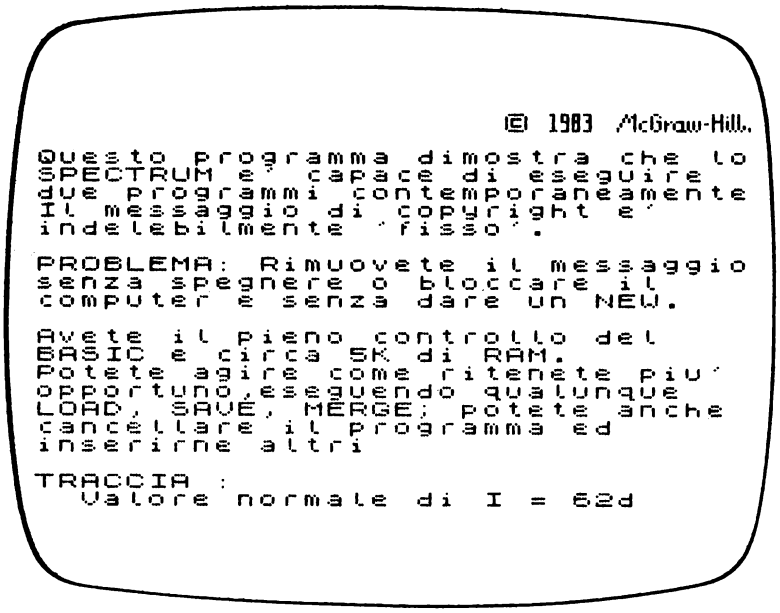
Linea		Linea		Linea		Colonna																										
						0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	4000		4700		5000	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
0	4100	8	4800	16	5100																																
	4200		4900		5200																																
	4300		4A00		5300																																
	4400		4B00		5400																																
	4500		4C00		5500																																
	4600		4D00		5600																																
	4700		4E00		5700																																
	4020		4720		5020																																
	4120		4820		5120																																
	4220		4920		5220																																
1	4320	9	4A20	17	5320																																
	4420		4B20		5420																																
	4520		4C20		5520																																
	4620		4D20		5620																																
	4720		4E20		5720																																
	4040		4740		5040																																
	4140		4840		5140																																
	4240		4940		5240																																
2	4340	10	4A40	18	5340																																
	4440		4B40		5440																																
	4540		4C40		5540																																
	4640		4D40		5640																																
	4740		4E40		5740																																
	4060		4760		5060																																
	4160		4860		5160																																
	4260		4960		5260																																
3	4360	11	4A60	19	5360																																
	4460		4B60		5460																																
	4560		4C60		5560																																
	4660		4D60		5660																																
	4760		4E60		5760																																
	4080		4780		5080																																
	4180		4880		5180																																
	4280		4980		5280																																
4	4380	12	4A80	20	5380																																
	4480		4B80		5480																																
	4580		4C80		5580																																
	4680		4D80		5680																																
	4780		4E80		5780																																
	40A0		47A0		50A0																																
	41A0		48A0		51A0																																
	42A0		49A0		52A0																																
5	13A0	13	4AA0	21	53A0																																
	44A0		4BA0		54A0																																
	45A0		4CA0		55A0																																
	46A0		4DA0		56A0																																
	47A0		4EA0		57A0																																
	40C0		47C0		50C0																																
	41C0		48C0		51C0																																
	42C0		49C0		52C0																																
6	43C0	14	4AC0	22	53C0																																
	44C0		4BC0		54C0																																
	45C0		4CC0		55C0																																
	46C0		4DC0		56C0																																
	47C0		4EC0		57C0																																
	40E0		47E0		50E0																																
	41E0		48E0		51E0																																
	42E0		49E0		52E0																																
7	43E0	15	4AE0	23	53E0																																
	44E0		4BE0		54E0																																
	45E0		4CE0		55E0																																
	46E0		4DE0		56E0																																
	47E0		4EE0		57E0																																

Appendice

Programma di commiato



Lo scopo di questo programma, l'ultimo di quelli contenuti nella cassetta *ZX Spectrum Arcade Routines,* è quello di dimostrare che lo Spectrum è in grado di eseguire una routine in codice macchina pur lasciando all'utente il pieno controllo del BASIC.
Caricato il programma, il computer mostrerà un messaggio di copyright nell'angolo superiore destro dello schermo. Questo messaggio resterà sempre presente. Dovrete scoprire come è ottenuto e rimuoverlo senza spegnere il computer e senza eseguire un NEW.
Il programma è tale da rendere impossibile la produzione di un listato utilizzabile ed è dotato di varie caratteristiche che rendono difficile inseguire il codice macchina. Per rimuovere il messaggio avrete bisogno di una routine in codice macchina.
Se riuscite a rimuovere il messaggio ed a disassemblare la routine scoprirete nuovi orizzonti di programmazione in codice macchina.
Buona fortuna.



Programma PROBLEMA

```
  10 RANDOMIZE USR (PEEK (PEEK 23627+256*PEE
K 23628+1)+(256*(PEEK (PEEK 23627+256*PEEK 23
628+2))))
  20 CLEAR 29100
  30 PRINT AT 2,0;"Questo programma dimostra
che loSPECTRUM e' capace di eseguire  due pro
grammi contemporaneamenteIl messaggio di copy
right e'    indelebilmente 'fisso'. "
  35 PRINT
  40 PRINT "PROBLEMA: Rimuovete il messaggios
enza spegnere o bloccare il    computer e sen
za dare un NEW."
  50 PRINT
  60 PRINT "Avete il pieno controllo del    B
ASIC e circa 5K di RAM."'"Potete agire come r
itenete piu' opportuno,eseguendo qualunque
LOAD, SAVE, MERGE; potete anche cancellare il
 programma ed      inserirne altri"
  70 PRINT '"TRACCIA : "'"  Valore normale di
 I = 62d"
  80 STOP
  90 LOAD ""CODE : GO TO 10
```

Software

ZX Spectrum Machine Code Assembler

ISBN 88-7700-901-2

Un completo programma assemblatore indispensabile per compilare i propri programmi scritti in Assembler secondo le specifiche descritte nel libro di T. Woods, *L'Assembler per lo ZX Spectrum*. È il naturale complemento del libro e permette di ottenere dei programmi molto più veloci e compatti dei corrispondenti in BASIC.
La cassetta è corredata da un manuale d'uso in italiano.

ZX Spectrum Arcade Games Routines

ISBN 88-7700-903-9

La cassetta contiene tutti i programmi presentati in questo libro e fa quindi risparmiare ore e ore di lavoro sulla tastiera, eliminando il rischio degli errori e delle correzioni.
I programmi in codice macchina sono presenti sia già compilati, e quindi direttamente utilizzabili, sia in formato sorgente, cioè come istruzioni REM in BASIC.
In questo modo possono essere modificati e adattati dall'utente che può assemblarli agevolmente usando lo *ZX Spectrum Machine Code Assembler*.

Nella stessa serie:

- C.A. Street, *La gestione delle informazioni con lo ZX Spectrum*
- T. Woods, *L'Assembler per lo ZX Spectrum*
- J. Heilborn e R. Talbott, *Guida al Commodore 64*
- R. Jeffries, G. Fisher e B. Sawyer, *Divertirsi giocando con il Commodore 64*
- H. Mullish e D. Kruger, *Il BASIC Applesoft*
- H. Peckham, *Il BASIC e il PC-IBM in pratica*
- H. Peckham, *Il BASIC e il Commodore 64 in pratica*
- G. Bishop, *Progetti hardware con lo ZX Spectrum*
- S. Kamins e M. Waite, *Programmazione umanizzata in Applesoft*
- N. Williams, *Progettazione di giochi d'avventura con lo ZX Spectrum*
- A. Pennell, *Guida allo ZX Microdrive e all'Interface 1*

Di prossima pubblicazione:

- P. Cohen, *Grafica e animazione con gli Apple II*
- D. Duff, *Guida al Macintosh*
- G. Kane, *Il manuale MC 68000*
- P. Hoffman e T. Nicoloff, *Il manuale MS-DOS*

Gli utenti dello Spectrum che hanno già una buona conoscenza dell'Assembler troveranno in questo libro lo strumento ideale per perfezionarsi; esso infatti approfondisce la teoria del linguaggio e ne presenta numerose applicazioni: grafica ad alta risoluzione, movimento di figure e di sfondi, rilevatori di collisione, contatori veloci, uso avanzato del colore e del suono e molte altre ancora.
Tutte queste tecniche sono ben documentate e immediatamente applicabili in quei programmi - come i video giochi - nei quali grafica di alta qualità, colore, suono, movimento e velocità sono elementi fondamentali.
In appendice sono presentati i listati di alcune routine molto potenti e veloci che possono sostituire le corrispondenti incorporate nella ROM.
Tecniche avanzate in Assembler con lo ZX Spectrum è quindi il naturale complemento del volume introduttivo di Tony Woods, **L'Assembler per lo ZX Spectrum**, pubblicato nella stessa collana.
Tutti i programmi in codice macchina contenuti nel libro sono stati scritti facendo uso dello **ZX Spectrum Machine Code Assembler**, disponibile su cassetta.

Lire 18 000
(IVA 2% inclusa)

ISBN 88 7700 010 4

S. Nicholls **Tecniche avanzate in Assembler con lo ZX Spectrum** McGraw-Hill